

Quantum Cybersecurity Red Team Guide

A hands-on textbook for Cybertection Training Center students

Cybertection LLC

Draft v0.2, October 2026

How to Use This Book

This textbook teaches security staff how quantum computing changes offensive and defensive security work. You will run real experiments, not just read about them. Every chapter ends with a lab in the Cybertection Quantum Labs hub, and most labs run on IonQ's simulator at no cost to you.

Each chapter follows the same pattern so you always know where you are:

Big idea: the one thing to remember from the chapter.

Plain-language explanation: the concept without jargon.

Worked example: the concept with numbers, step by step.

Your turn: the lab, and what to record.

Check questions: test yourself before you move on.

Work through the book in order. Parts 1 and 2 give you the foundations and the Kali skills that Parts 3 to 5 depend on. If a term is unfamiliar, check Appendix B.

Who this book is for: Training Center students preparing for red team and SOC roles, and anyone who needs to understand quantum risk to real systems. You do not need a physics background. You do need comfort with basic networking and the command line.

What you need: a web browser for the browser labs, and a Kali Linux environment (your own, or the one provided) for the Kali labs.

Authorized Use and Safety Rules

Read these rules before any lab. They apply to every chapter.

1. **Attack only Cybertection lab targets.** Scans, captures, floods, and brute-force tests run only against machines inside the Cybertection lab network. Never run them against a network, site, or person you do not have written permission to test.
2. **Use synthetic or lab-captured data only.** Do not capture traffic from a real network, and do not use real customer or employee data in a lab.

3. **No hardware runs for students.** Labs run on the local backend or the IonQ simulator. Real quantum hardware runs are instructor-only, because they cost credits.
4. **Never share or commit an API key.** The lab platform holds the key. If you think a key has been exposed, tell your instructor immediately.
5. **Keep evidence clean.** Save your outputs in the folder structure from Chapter 5. Your report must show what you did and why.

Why these rules matter: security work is valuable only when it is authorized. A scan that looks harmless can take down a fragile server, and an unauthorized scan can be a crime even when you meant well. Every skill in this book is one you will use on a real job, and the habits you build here, such as getting written scope and keeping evidence, are the ones employers check first.

Breaking these rules ends your access to the labs and may breach the law. The rules are part of the training.

Part 1: Quantum Foundations

Chapter 1: Why Quantum Matters to Security Staff

Big idea: quantum computers do not simply try every answer at once. They are good at one specific trick, and that trick breaks some of the math that protects today's data.

Plain-language explanation

A **bit** is a switch that is either off (0) or on (1). Every computer you have ever used is built from billions of these.

A **qubit** is the quantum version. Its state can be a *mix* of 0 and 1 at the same time, called **superposition**. A helpful way to picture it: a coin spinning in the air is neither heads nor tails until it lands. The catch is that when you look at a qubit, it lands. You get one answer, either 0 or 1, and the mix is gone. This is **measurement**, and it is destructive.

Entanglement is a link between two or more qubits. If you measure one entangled qubit and get 0, the other is guaranteed to give a matching result when you measure it. The two results stay connected even when they are far apart. This does not let you send messages faster than light, but it makes quantum systems behave in ways classical computers cannot copy.

Quantum computers use **interference**. Each possible answer adds or cancels with the others, like waves on water. A good algorithm arranges things so that wrong answers cancel out and the right answer gets stronger. When you finally measure, you are likely to see the right answer.

Worked example: one qubit in a Hadamard gate

Start with a qubit in state 0. Apply a **Hadamard gate**, which puts it into an equal mix of 0 and 1.

Step	State	Chance of measuring 0	Chance of measuring 1
Start	0	100%	0%
After Hadamard	equal mix	50%	50%
Measure 10 times	(each measurement lands on one value)	about 5 zeros	about 5 ones

Measuring ten times gives roughly half zeros and half ones, but each single run gives one answer. The quantum power comes from designing circuits where the wrong answers cancel before you measure.

Why this threatens security

Much of today's encryption depends on problems that are hard for classical computers. Two examples:

- **RSA** depends on the difficulty of factoring a large number into primes.
- **Elliptic-curve cryptography** depends on the difficulty of a certain kind of math problem on curves.

A large, fault-tolerant quantum computer running **Shor's algorithm** can solve both. Today's machines are far too small for this. The threat is about the future, and you will see why in Chapter 12.

Key takeaway: superposition, entanglement, and interference let quantum computers solve some specific problems much faster. Measurement gives you only one answer per run, so good algorithms use interference to make the right answer likely.

Your turn: Lab B0. Read the lab lessons in the hub and answer the check questions. Save your answers.

Check questions 1. What happens to a qubit's superposition when you measure it? 2. Why does a quantum computer not give you every answer at once? 3. Why can a quantum computer break RSA in principle, even though it cannot today?

Chapter 2: Quantum Machine Learning for Detection

Big idea: a quantum model compares data points using the overlap between quantum states, and that comparison is only useful if the data is encoded well.

Plain-language explanation

Imagine you want to detect a port scan from network traffic. You measure four things about each traffic window:

1. **Packet rate:** how many packets per second.

2. **Bytes ratio:** how many bytes go out compared with how many come in.
3. **Port entropy:** how spread out the destination ports are. A scan hits many ports, so its entropy is high.
4. **SYN ratio:** how many packets are connection requests compared with the total.

A classifier learns the boundary between “normal” and “scan” from labeled examples. A **support vector machine (SVM)** does this by finding the widest gap between the two groups. A **kernel** is a function that measures how similar two points are, and an SVM uses that similarity to draw curved boundaries.

A **quantum kernel** works the same way, but the similarity is computed from quantum states. A **feature map** is the circuit that turns your four numbers into a quantum state. The model is **QSVC**, a quantum SVM.

Worked example: encoding one traffic window

Suppose one window has these normalized features, each scaled between 0 and 1:

Feature	Value
Packet rate	0.80
Bytes ratio	0.30
Port entropy	0.90
SYN ratio	0.75

The feature map rotates qubits by angles based on these values, then entangles the qubits so that features interact. Two windows that look alike produce similar quantum states, so their overlap is high. A scan window and a normal window produce states with low overlap.

Important: more qubits and more layers do not guarantee better accuracy. Each extra layer adds noise on real hardware and can make the model harder to train.

Comparing the two models

	Classical SVM (baseline)	Quantum kernel SVM (QSVC)
Kernel	RBF, computed on a laptop	Fidelity kernel, computed on a quantum circuit
Speed on this lab	Milliseconds	Seconds to minutes
Accuracy	Often strong on small, clean data	Depends on encoding and data

Worked example of a misleading result: suppose the test set has 6 samples. The classical model gets 6 of 6 right (100%) and the quantum model gets 4 of 6 right (67%). That looks like a big gap, but one mistake is 17 percentage points. With six samples, one different sample could reverse the result. Always compare on at least three seeds and a larger test set.

Key takeaway: a quantum classifier is a different way to measure similarity. Judge it by fair comparison, not by one lucky run.

Your turn: Lab B1, Quantum Threat Detection. Run the classical SVM and QSVC on synthetic traffic. Record the accuracy and training time for each run. Then repeat with seeds 1, 2, and 3.

Your turn: Lab B2, Feature map design. Change how the four features are encoded. Record which encoding gives the best accuracy and how much training time it costs.

Check questions 1. Why might a 6-sample test set give a misleading accuracy? 2. What is the difference between a kernel and a feature map? 3. What would you change to test the quantum model fairly?

Chapter 3: Quantum Search, Randomness, and Key Exchange

Big idea: quantum methods can search faster, generate true randomness, and detect eavesdropping. Each one has a security meaning you need to understand.

3a. Grover's search

Plain-language explanation. Suppose you have a phone book with N names and no index, and you need to find one name. A classical computer checks names one by one, about $N/2$ checks on average. **Grover's algorithm** does better: it uses interference to make the correct answer more likely with each step. It needs about the square root of N steps.

This is not the same as cracking a password instantly. Grover's speedup is quadratic, and it works best on problems where you can check an answer quickly.

Worked example: four items. Let the marked item be item 3 out of four.

Step	What happens	Result
Start	All four items equally likely	25% each
One Grover step	The marked item's chance is boosted	Item 3: 100%
Measure	You read the answer	Item 3

For $N = 4$, one Grover step is enough, so the classical method needs about two checks on average while Grover needs one. The gap grows with N , but for four items the example shows the mechanism clearly.

Security meaning. Grover's algorithm roughly halves the effective strength of a symmetric key. A 128-bit key behaves like a 64-bit key against a quantum search. That is why security guidance often recommends 256-bit symmetric keys for long-lived data.

3b. Quantum random numbers

Plain-language explanation. Computers generate random numbers using formulas, so their output is predictable if an attacker knows the formula and the starting value. That is **pseudo-**

randomness. A quantum measurement is random in a deeper sense: the outcome is not determined in advance by any hidden value you could find.

Worked example. Put a qubit in an equal superposition and measure it. Each measurement gives 0 or 1 with equal chance. Repeat 1,000 times and you should see close to 500 zeros. Testing that number (and many others) is how you check the output.

Security meaning. Good keys need good randomness. If the random source is biased, an attacker can guess part of the key without breaking the cipher. This is why key generation is one of the most important parts of a security system.

3c. BB84 key exchange

Plain-language explanation. BB84 lets two people, Alice and Bob, agree on a secret key over a channel that an eavesdropper, Eve, may be listening to.

- Alice sends single photons, each encoded as a bit in one of two **bases**: rectilinear (+) or diagonal (×).
- Bob measures each photon in a basis he picks at random.
- They publicly compare bases, not bits, and keep only the bits where the bases matched. This is **sifting**.
- They sample some bits to estimate the **quantum bit error rate (QBER)**. A high error rate means someone measured the channel.

Worked example: eight bits, no eavesdropper.

Position	1	2	3	4	5	6	7	8
Alice's bit	1	0	1	1	0	0	1	0
Alice's basis	+	×	×	+	+	×	+	×
Bob's basis	+	×	+	+	×	×	+	×
Match?	yes	yes	no	yes	no	yes	yes	yes

Sifting keeps positions 1, 2, 4, 6, 7, and 8. With no eavesdropper, Bob's bits match Alice's at every kept position, so QBER is 0%.

Worked example: with Eve. Eve intercepts each photon, measures it in a random basis, and resends it. When Eve guesses the wrong basis, she randomizes the photon, so Bob's bit is wrong half the time. Eve guesses the wrong basis half the time, so about 25% of the sifted bits are wrong. A 25% QBER is a strong signal that someone is listening.

Key takeaway: BB84's security comes from the fact that measuring a quantum state changes it. An eavesdropper always leaves a trace, and the error rate reveals it.

Your turn: Lab B5, Grover's search. Find the marked item among four options. Record the number of tries Grover needs and compare it with a classical search.

Your turn: Lab B6, Quantum random numbers. Generate random bits and run the statistical tests. Write one paragraph on why key generation needs good randomness.

Your turn: Lab B7, BB84 key exchange. Run the simulation with and without Eve. Record the QBER for each case and confirm it is close to 0% without Eve and close to 25% with Eve.

Check questions 1. Why does an intercept-and-resend attack raise the error rate in BB84? 2. Grover's algorithm needs about the square root of N steps. What does that mean for a 128-bit symmetric key? 3. What would a biased random number generator do to a key?

Chapter 4: Shor's Algorithm and the Threat to RSA

Big idea: RSA is secure because factoring large numbers is hard. Shor's algorithm makes factoring fast on a large quantum computer.

Plain-language explanation

RSA uses a public key built from two large prime numbers, p and q . Anyone can see their product, $n = p \times q$. Breaking RSA means finding p and q from n . For a 2048-bit key, that takes classical computers far longer than the age of the universe.

Shor's algorithm turns factoring into a **period-finding** problem. It finds the period of a function that depends on the number you want to factor. Quantum interference reveals that period efficiently. Once you have the period, a classical calculation gives you the factors.

Worked example: factoring 15

The circuit in Lab B8 factors 15. Here is the logic step by step, using the number 15 instead of a quantum circuit:

1. Pick a random number a smaller than 15, say $a = 7$.
2. Find the period r where $7^r \bmod 15$ returns to 1. The values are 7, 4, 13, 1, so $r = 4$.
3. Since r is even, compute $7^{(4/2)} - 1 = 48$ and $7^{(4/2)} + 1 = 50$.
4. Take the greatest common divisors: $\gcd(48, 15) = 3$ and $\gcd(50, 15) = 5$.
5. The factors of 15 are 3 and 5.

Shor's quantum step does the period-finding in step 2. The rest is classical arithmetic.

Why this does not break RSA today

The circuit in Lab B8 uses a handful of qubits to factor 15. A real RSA-2048 break would need a much larger, error-corrected quantum computer. Factoring 15 is a demonstration of the mechanism, not a threat to any real key.

Why it still matters now

Data encrypted today with RSA may be stored by an attacker and decrypted later, once a large quantum computer exists. This is **harvest now, decrypt later**, covered in Chapter 12.

Key takeaway: Shor's algorithm turns factoring into period-finding, and the quantum computer does the period-finding. The threat is real, but it depends on machines that do not exist yet.

Your turn: Lab B8, Shor's algorithm on 15. Run the circuit and record the factors it finds. Then explain in your own words why the classical steps still matter.

Check questions 1. If you had picked $a = 5$ instead of 7, what would $\gcd(5, 15)$ tell you immediately, and why would you not need period-finding? 2. Why does a small demonstration not show that RSA is broken today? 3. Which part of Shor's algorithm needs a quantum computer?

Part 2: Recon and Capture (Kali)

Chapter 5: Kali Basics and Evidence

Big idea: a finding is only as good as the evidence behind it. A result you cannot reproduce is not a result.

Plain-language explanation

Kali Linux is a Debian-based system with penetration testing tools already installed. In this course you will use it for scanning, capture, and analysis.

Good evidence has three properties:

- **Organized:** every lab has its own folder.
- **Timestamped:** you know when each step ran.
- **Reproducible:** someone else could run the same commands and get the same result.

Recommended folder structure

```
lab-K2-port-scan/
├── commands.txt      # every command you ran, in order
├── raw/              # raw tool output (nmap, pcap files, logs)
├── notes.md          # what you saw and what you think it means
└── screenshots/     # optional, labeled by step
```

Worked example: saving evidence

Instead of this:

```
nmap something
```

Do this, so you can prove what you ran and when:

```
date >> commands.txt
echo "nmap -sV -oN raw/nmap_scan.txt 10.0.0.5" >> commands.txt
nmap -sV -oN raw/nmap_scan.txt 10.0.0.5
```

The `-oN` flag saves the output to a file, and the `date` line records when you started.

Your turn: Lab K1, Kali basics for pentesting. Set up your Kali environment, create the folder structure, and save a log that shows the setup works.

Check questions 1. Why is `commands.txt` as important as the scan output? 2. What would someone need to reproduce your result?

Chapter 6: Port Scanning

Big idea: a port scan shows what is listening on a machine. It tells you what to investigate next, not whether the machine is safe.

Plain-language explanation

A computer has 65,535 TCP ports. Services listen on some of them. Web servers usually use port 80 or 443, and SSH uses port 22. A **port scan** sends a probe to each port and records the response. An open port answers. A closed port refuses. A filtered port does not respond at all, often because a firewall dropped the probe.

Nmap is the standard tool. Its `-sV` option also asks each open port which service and version is running, which helps you find out-of-date software.

Worked example: reading scan output

Suppose the scan of a lab target returns:

Port	State	Service	Version
22	open	ssh	OpenSSH 8.2
80	open	http	Apache 2.4.41
3306	filtered	mysql	(no response)

What you can say: the host runs SSH and a web server, and the database port is hidden from you.

What you cannot say yet: whether Apache 2.4.41 has a vulnerability. That is a separate question. The scan gives you a lead, and your next step is to check the version against known issues, inside the lab only.

Your turn: Lab K2, Port scanning. Scan the Cloud Labs target, save the output with `-oN`, and list the open services with their versions.

Check questions 1. What does a “filtered” port tell you that a “closed” port does not? 2. Why does the version number matter more than the open port number?

Chapter 7: Traffic Capture and Feature Extraction

Big idea: a classifier learns from examples. Your labels are the most important part of the dataset.

Plain-language explanation

A **packet capture (pcap)** file records every packet a network interface sees. Wireshark and tcpdump both create them. In this course you capture two sessions:

1. **Normal:** a browsing session on the lab network.
2. **Scan:** an Nmap scan against the lab target.

Then you turn each capture into four numbers, the features from Chapter 2. The labeled rows become your training data.

The four features in plain terms

Feature	What it measures	Why it helps
Packet rate	packets per second	Scans and floods send bursts
Bytes ratio	outgoing bytes divided by incoming bytes	Scans send small packets and get little back
Port entropy	how spread out the destination ports are	Scans touch many different ports
SYN ratio	connection-start packets divided by all packets	Scans open many half-finished connections

Worked example: one row of the CSV

label	packet_rate	bytes_ratio	port_entropy	syn_ratio
normal	42	1.8	0.21	0.05
scan	310	0.4	0.92	0.71

The scan row has a higher packet rate, a lower bytes ratio, higher port entropy, and a much higher SYN ratio. A classifier can learn that pattern from many rows like these.

Labeling mistakes that ruin a dataset

- **Swapped labels:** if you label the scan capture as normal, the model learns the wrong pattern.
- **Mixed captures:** if browsing traffic leaks into the scan capture, the labels are no longer true.
- **Too little data:** one capture per class is not enough. Capture more than one session when you can.

Your turn: Lab K3, Traffic capture. Save two pcap files with clear names, such as `normal_2026-10-10.pcap` and `scan_2026-10-10.pcap`.

Your turn: Lab K4, Feature extraction. Run the provided script on each pcap. Check that the CSV has one row per capture and that the labels match your file names.

Check questions 1. Why does a port scan raise the SYN ratio? 2. What could go wrong if the normal and scan captures were mixed up? 3. Why is one capture per class not enough to train a model?

Part 3: Quantum-Assisted Detection

Chapter 8: Scan Detection with a Quantum Classifier

Big idea: train on your own labeled data, then judge the model by what it gets wrong, not only by its headline accuracy.

Plain-language explanation

The model you built in Chapter 2 is now trained on your features from Chapter 7. You will upload the CSV, train the quantum classifier, and read the predictions.

Reading a confusion matrix

A **confusion matrix** shows every kind of result:

	Predicted normal	Predicted scan
Actual normal	9 (true negative)	1 (false positive)
Actual scan	2 (false negative)	8 (true positive)

From this table:

- **Accuracy** = $(9 + 8) / 20 = 85\%$.
- **Missed scans** = 2 false negatives. In security, these are often the costliest mistakes.
- **False alarms** = 1 false positive. Too many of these cause alert fatigue for analysts.

Accuracy alone can hide the difference. A model that says “normal” every time gets 50% on a balanced test set, and it catches zero scans.

Your turn: Lab K5, Scan detection. Upload your K4 features, train the classifier, and record the accuracy, the confusion matrix, and the prediction for each capture.

Check questions 1. In the example, how many scans did the model miss? 2. Why is a model that always says “normal” a bad detector even at 50% accuracy?

Chapter 9: SYN Floods and Brute-Force Attacks

Big idea: each attack leaves a different traffic signature. Learn the signature first, then build the dataset.

SYN floods

Plain-language explanation. A normal TCP connection starts with a SYN packet, the server replies with SYN-ACK, and the client finishes with ACK. A **SYN flood** sends SYN packets and never finishes the handshake. The server keeps half-open connections in memory until its table is full, and then it cannot accept real users.

Worked example. A server with a 128-entry backlog gets 500 SYN packets in one second and no ACKs. The backlog fills in under a second, and legitimate connection attempts are dropped until entries expire.

Signature to look for: a high packet rate, a very high SYN ratio, and many half-open connections with few replies.

Brute-force attacks

Plain-language explanation. A brute-force attack tries many passwords against a login page. Tools like Hydra automate it. Each failed attempt is a login error, so the signature is a burst of failed logins from one source.

Worked example. A login page logs 60 failed attempts in 30 seconds from one IP address, followed by a success. The burst is the signal. A single failed login is normal.

Signature to look for: many failed logins in a short window, with a repeated username or source.

Why labs use isolated targets

A SYN flood can stop a real server. Lab VMs are isolated so that the flood affects only the target you chose, and so your results are safe to repeat.

Your turn: Lab K6, SYN flood detection. Create the flood against the isolated lab VM, capture it, and train a classifier on the flood and normal captures.

Your turn: Lab K7, Brute-force detection. Capture the failed-login bursts on the lab login page and train a classifier on them.

Check questions 1. Why does a SYN flood fill the server's backlog? 2. What makes a brute-force burst different from a few normal failed logins? 3. Why should lab floods run only on isolated machines?

Chapter 10: Log Analysis and Model Comparison

Big idea: a fair comparison needs the same data, the same split, and several metrics, not one accuracy number.

Plain-language explanation

Logs and captures come from many sources. Before you train a model, you parse them into a single table, label each row, and split the table into training data and test data. The model learns from the training rows and is judged on the test rows it has never seen.

Metrics beyond accuracy

Metric	Formula	Meaning
Precision	$\text{true positives} / (\text{true positives} + \text{false positives})$	When it says “scan,” how often it is right
Recall	$\text{true positives} / (\text{true positives} + \text{false negatives})$	Of all real scans, how many it found
F1 score	$2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$	One number that balances the two

Worked example. Using the confusion matrix from Chapter 8: precision = $8 / (8 + 1) = 89\%$, and recall = $8 / (8 + 2) = 80\%$. The model finds 80% of scans, and 89% of its scan alerts are real.

A fair comparison checklist

- Same training and test rows for both models.
- Several random seeds, not one.
- A test set large enough that one sample cannot flip the conclusion.
- Report time as well as accuracy. A model that is 2% more accurate but ten times slower may not be worth it.

Your turn: Lab K8, Log analysis and model comparison. Build the labeled dataset from your attack logs, run both models on the same split, and produce a comparison table with accuracy, precision, recall, and training time. Write a short conclusion.

Check questions 1. Which metric matters most when missing an attack is costly? Why? 2. Why must both models use the same split? 3. How many seeds would you run before trusting the comparison, and why?

Part 4: Crypto and Quantum Risk

Chapter 11: TLS Audit

Big idea: an audit tells you which algorithms a system uses today, and that list is the starting point for quantum risk.

Plain-language explanation

TLS protects web traffic. When your browser connects to a site, the server and browser agree on a **key exchange** algorithm, a **certificate** with a signature algorithm, and a **cipher** for encrypting the data. Each of these choices has a different quantum risk.

Part	Common example	Quantum risk
Key exchange	ECDHE, RSA key transport	High: broken by Shor's algorithm
Certificate signature	RSA or ECDSA	High: forged by Shor's algorithm
Symmetric cipher	AES-256	Lower: Grover only halves its strength

Worked example: reading a finding

A testssl.sh or sslscan run on a lab server reports:

Key exchange: ECDHE with X25519
Certificate: RSA 2048
Cipher: AES-256-GCM

- The key exchange is vulnerable to a future quantum computer, so it is a **high** quantum-risk finding.
- The RSA 2048 certificate is also high risk, because Shor's algorithm breaks RSA.
- AES-256 is acceptable, because Grover only reduces its strength to about 128-bit.

Your turn: Lab K9, TLS audit. Run sslscan or testssl.sh on the lab server, record each finding, and rate it by severity and quantum risk.

Check questions 1. Why is a 256-bit symmetric cipher less exposed to quantum attacks than RSA? 2. In the example, which finding would you fix first, and why?

Chapter 12: Harvest-Now-Decrypt-Later Risk

Big idea: the question is not “can a quantum computer break this today?” It is “how long must this data stay secret?”

Plain-language explanation

An attacker who records encrypted traffic today cannot read it yet. If a quantum computer later breaks the key exchange, the recorded traffic can be decrypted. This is **harvest now, decrypt later**.

The risk depends on two things:

1. **Secrecy period:** how long the data must stay confidential. Medical records and trade secrets may need decades. A password reset link needs minutes.

2. **Migration time:** how long it takes to switch to a quantum-safe algorithm. Large systems take years.

If the secrecy period is longer than the migration time plus the time until a quantum computer exists, the data is already at risk.

Worked example: ranking systems

System	Secrecy period	Priority
Medical records (archived traffic)	30+ years	Highest
Customer payment data	5 to 10 years	High
Internal chat logs	1 to 2 years	Medium
Password reset links	Minutes	Low for this risk

The ranking tells you where to start. The top of the list needs quantum-safe protection first.

Your turn: Lab B9, Quantum risk assessment. Answer the scenario questions and rank the systems. Explain each ranking in one sentence.

Your turn: Lab K10, Harvest-now-decrypt-later assessment. Use your K9 findings to rank the lab's systems by how long their data must stay secret. Use the same method as the table above.

Check questions 1. What two factors decide whether data is at risk from harvest-now-decrypt-later? 2. Why might a password reset link be low priority for this risk, even though it is sensitive?

Chapter 13: Post-Quantum Migration

Big idea: migration is a process, not a switch. You inventory, prioritize, test, and then roll out.

Plain-language explanation

Post-quantum cryptography (PQC) uses algorithms based on math problems that quantum computers are not known to solve efficiently. In 2024, NIST published its first standards:

- **ML-KEM (FIPS 203):** for key exchange.
- **ML-DSA (FIPS 204):** for digital signatures.
- **SLH-DSA (FIPS 205):** a second signature standard, based on hash functions.

Many systems use **hybrid key exchange**, which combines a classical algorithm with a post-quantum one. If the new algorithm has a flaw, the classical one still protects the session, and if a quantum computer appears, the post-quantum one still protects it.

Migration steps

1. **Inventory:** list every place a system uses RSA, ECDSA, or ECDHE.
2. **Prioritize:** use the ranking from Chapter 12.

3. **Test:** enable hybrid mode on a test server and check compatibility.
4. **Roll out:** update the highest-priority systems first, then the rest.
5. **Monitor:** check that clients still connect and that performance is acceptable.

Worked example: a migration note

System: internal web server, lab target. **Current:** ECDHE key exchange, RSA 2048 certificate. **Risk:** high, because recorded traffic could be decrypted later and the certificate could be forged. **Change:** enable hybrid X25519 with ML-KEM for key exchange when the server supports it, and plan a certificate update to a post-quantum signature when supported by the certificate authority. **Owner and date:** [to be filled in by the team].

Your turn: Lab K11, Post-quantum migration check. Check the lab server for hybrid key exchange support and write a migration note in the format above.

Check questions 1. Why is hybrid key exchange safer during the transition than switching fully to a new algorithm? 2. What are the five steps of migration, and which one is most often skipped?

Part 5: Reporting and Capstone

Chapter 14: Findings Report

Big idea: a report is how your work gets acted on. A finding nobody understands does not get fixed.

Plain-language explanation

Every finding in your report needs four things: what you found, the evidence, how serious it is, and what to do about it. Write for two readers: a technical reader who wants the commands, and a manager who wants to know what matters.

Severity and quantum risk

Use two ratings:

- **Severity:** how bad the issue is today (low, medium, high, critical).
- **Quantum risk:** how exposed the issue is to a future quantum attacker (low, medium, high).

A finding can be low severity and high quantum risk. For example, an old RSA certificate may work fine today but be a high quantum-risk item.

Report structure

1. **Summary:** one page, for a non-technical reader.
2. **Scope and authorization:** the targets, the dates, and who approved the test.

3. **Findings:** each with evidence, severity, and quantum-risk rating.
4. **Recommendations:** in priority order, with an owner for each.
5. **Appendix:** commands and raw output.

Worked example: one finding

Finding 2: RSA 2048 certificate on the lab web server. **Severity:** medium. The certificate works today. **Quantum risk:** high. A future quantum computer could forge the signature. **Evidence:** testssl.sh output, saved in raw/testssl_web.txt. **Recommendation:** plan a migration to a post-quantum signature when the certificate authority supports it. Owner: [name]. Target: [date].

Your turn: Lab K12, Findings report. Write up findings from K2 to K11 in the format above and submit the report through the hub.

Check questions 1. Why rate severity and quantum risk separately? 2. Who is the summary written for, and what does it leave out?

Chapter 15: Capstone

Big idea: the capstone combines every skill in the book on one engagement.

What the capstone asks you to do

1. **Recon:** scan the Cloud Labs target and map its services (Chapter 6).
2. **Capture:** record normal and attack traffic (Chapter 7).
3. **Detect:** train and judge a quantum classifier on your data (Chapters 8 to 10).
4. **Audit:** check the TLS configuration and rank the quantum risk (Chapters 11 to 13).
5. **Report:** write the findings report with severity and quantum-risk ratings (Chapter 14).

How you will be graded

Area	What a strong answer shows
Authorization	Scope is written down and every action stays inside it
Evidence	Commands and outputs are saved and reproducible
Analysis	Models are compared fairly, with metrics and seeds
Quantum risk	Findings are ranked by how long the data must stay secret
Report	Clear for both a technical reader and a manager

Your turn: Lab K13, Capstone. Complete the engagement on the Cloud Labs machine and submit your report and detection results through the hub.

Appendix A: Lab Index

Lab	Title	Chapter
B0	Quantum basics for security staff	1
B1	Quantum Threat Detection	2
B2	Feature map design	2
B5	Grover's search	3
B6	Quantum random numbers	3
B7	BB84 key exchange	3
B8	Shor's algorithm on 15	4
B9	Quantum risk assessment	12
K1	Kali basics for pentesting	5
K2	Port scanning	6
K3	Traffic capture	7
K4	Feature extraction	7
K5	Scan detection	8
K6	SYN flood detection	9
K7	Brute-force detection	9
K8	Log analysis and model comparison	10
K9	TLS audit	11
K10	Harvest-now-decrypt-later assessment	12
K11	Post-quantum migration check	13
K12	Findings report	14
K13	Capstone	15

B3 (noise on hardware) is instructor-only and is not in this book.

Appendix B: Glossary

- **Authorization:** written permission to test a specific target.

- **BB84:** a quantum key exchange protocol that detects eavesdroppers.
- **Confusion matrix:** a table of correct and incorrect predictions by class.
- **Entanglement:** a link between qubits so their results are correlated.
- **Feature map:** a circuit that encodes classical data into quantum states.
- **Grover's algorithm:** a quantum search method that needs about the square root of N steps.
- **Harvest now, decrypt later:** capturing encrypted data today to decrypt once quantum computers can break its key.
- **Hybrid key exchange:** combining a classical and a post-quantum algorithm during migration.
- **Kernel:** a measure of similarity between two data points.
- **Measurement:** reading a qubit's state, which gives one result and destroys the superposition.
- **Packet capture (pcap):** a file recording network packets.
- **Period finding:** the step in Shor's algorithm that a quantum computer performs.
- **Post-quantum cryptography:** algorithms designed to resist quantum attacks.
- **Precision and recall:** metrics for how many alerts are right and how many attacks are found.
- **QBER:** quantum bit error rate, the error rate in a BB84 key exchange.
- **Qubit:** the basic unit of quantum information.
- **SYN flood:** an attack that sends connection requests and never completes them.
- **Superposition:** a qubit's combination of 0 and 1 before measurement.

Appendix C: References

[References to be added before publication. Include the NIST post-quantum standards (FIPS 203, 204, 205), the BB84 paper, the Shor and Grover papers, and the IonQ documentation for the simulator.]