3-day class

Reverse Engineering Quantum Malcode & Investigating Quantum Computing Platforms

Instructor

Gary Golomb

Prepared for:

DEF CON 2027 Training Selection Committee

Table of Contents

Notes 4

Teaching approach 4

Bespoke reverse engineering software 4

Labs 4

Terms used in this document 5

Razorwire 5

Gate Crasher 5

Spalax 5

QC 5

Day 1: Reverse Engineering Quantum Code 6

Quantum Computing in the enterprise: How real and how common is it? 6

Why will quantum be hacked? 6

Getting familiar QC code using visualization tooling 6

Visualizing qubit interactions across code 8

Identifying interesting qubit operation patterns 10

Exploring gates and the math behind qubit operations 11

Understanding matrices as a map of input vs output basis 11

Superposition and Entanglement 12

Phase and Amplitude 13

Workshop: Phase, superposition, and entanglement 14

Day 1 LAB: Reverse engineering an unknown QC program 14

Day 2: Threats Against Quantum Platforms 16

Review of important concepts from the previous day 16

Day 2 Morning: Learning about the quantum development stack by understanding the threats at each layer 16

Classical compute components 16

Endpoint components 16

Cloud components 17

QPU Control Systems 17

QPU control electronics (quick overview) 17
Cryogenic enclosure (quick overview) 17
QPUs (deeper overview) 17
LAB: Writing a quantum application and examining the artifacts the dev stack creates . 18
LAB: Machine Learning-based approaches to circuit classification (detecting good vs bad circuits) 18
Day 2 Afternoon: Reverse engineering malicious QC code 18
Threats we'll do technical deep dives into 18
Hiding data in circuits 18
Circuit hijacking 18
Covert channels 18
Spalax: Exploiting opportunity in idle qubits 19
A survey of other published threats 19
GateCrasher: Polymorphic insertion and evasion attacks 20
LAB: Using GateCrasher to obfuscate a QC trojan and defeat ML detection models 20
Day 3 Morning 1.5 Hours - LAB: Reverse engineering an unknown malicious QC circuit (conditional trojan) 21
Day 3: Final Investigation and Reverse Engineering Project 21
Final project scoring 22

Notes

Teaching approach

Traditional approaches to teaching a subject such as manipulating quantum phenomena typically start with academic and theoretical fundamentals, then, after a significant length of time, synthesize lower-level principles into a higher-order working system. This course follows the hacker methodology of learning, which takes the opposite approach – that some people may remember from learning shellcoding in the 90s or PE and packed ASM analysis in the early 2000s – before these subjects were taught in computer science programs. In this course, we start with QASM (quantum assembly), learn to identify interesting patterns, analyze those patterns, and then learn the quantum fundamentals required to manipulate legitimate code and analyze malicious code.

Bespoke reverse engineering software

In the document that follows, you will see screenshots of in-house-developed tools that attendees will receive for free. Each tool is highlighted in the Terms section of this document and described throughout. While not publicly released yet, these tools have been in use on different platforms and have analyzed well over 100 different real-world quantum code projects. Public release dates are identified in the Terms section of this document.

Labs

Day 2 and 3 are heavy with hands-on labs. Day 1 will end with a lab for attendees to work through the day's concepts in aggregate. That means Day 1 will be dominated by teaching rather than lab work, but the lessons throughout the day will be highly interactive to keep attendees engaged. Prizes and giveaways usually accompany this style of teaching for me.

Terms used in this document

Razorwire

A tool I've been developing for reverse engineering quantum assembly. Think about it like IDA Pro, but for quantum. You'll see screenshots of it in this doc. It will be publicly released on GitHub in early 2027.

Gate Crasher

A tool I've been developing for generating polymorphic quantum circuits from a given input. It also implements "insertion and evasion" techniques, an idea first described in the 1998 paper "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" by Ptacek and Newsham – except, radically evolved for quantum ASM. Planned to be publicly released on GitHub mid-2027. Described a little more later in this doc.

Spalax

A tool I've been developing to embed malicious quantum circuits inside existing known-good circuits by identifying "slack space" in circuits - qubits that are unused by the legitimate code long enough to be hijacked for other uses, then reset before being used by the legitimate algorithm (or not reset, depending on the purpose of the inserted code). You'll see screenshots of it in this doc. Planned to be publicly released on GitHub mid-2027.

QC

Quantum Computing

Day 1: Reverse Engineering Quantum Code

Today focuses on how different quantum computers are from traditional classical computers. We start by examining real-world quantum code published by financial institutions and other researchers. We will reverse-engineer these algorithms, first by identifying patterns in the code, then learning what it takes to understand those patterns at increasingly lower levels.

Quantum Computing in the enterprise: How real and how common is it?

We'll get this question off the table immediately. People who have not yet worked with QC are typically surprised to see:

- How far along QC computing is.
- Which enterprises are already using it.
- How long some enterprises have been using it.
- The problems quantum can solve better than previous computing paradigms.
- When QC will be capable enough to get the same hype AI is getting today.

Why will quantum be hacked?

There's a standard, somewhat cliché way to prioritize defense cycles by assessing risk as a product of vulnerabilities and impact. At best, this leaves attackers and their motives opaque. At worst, and commonly, it doesn't understand the attacker at all.

The previous lessons showed how enterprises are using (and will be using) QC, now and in the near future. This brief, but important discussion contextualizes all the subsequent training by explaining who stands to gain most from attacking quantum systems (in the ways attendees will be learning), and why.

Getting familiar QC code using visualization tooling

We start by learning the differences between reverse engineering CPU disassembly vs examining QPU logic visually, starting with:

- Program Flow
    - Branching
        - Conditional logic and feedback

- Parallelism
  - Concurrent instructions and gates
  - Source lines vs operations vs layers

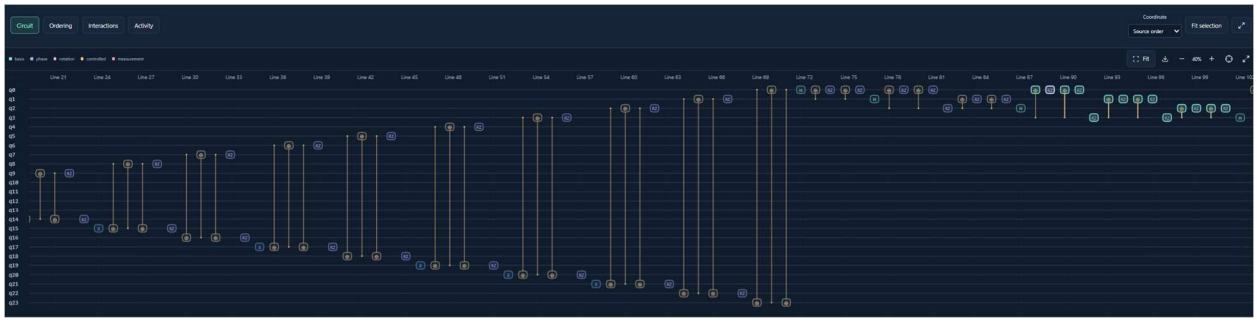


Figure: A screen from Razorwire showing a clearly discernible preamble in this program before a different pattern begins. This raises questions we will learn to answer with this tool, such as: What is this preamble doing? How does it set up later computation? Why do both sections look so different? What other types of patterns should we be looking for?

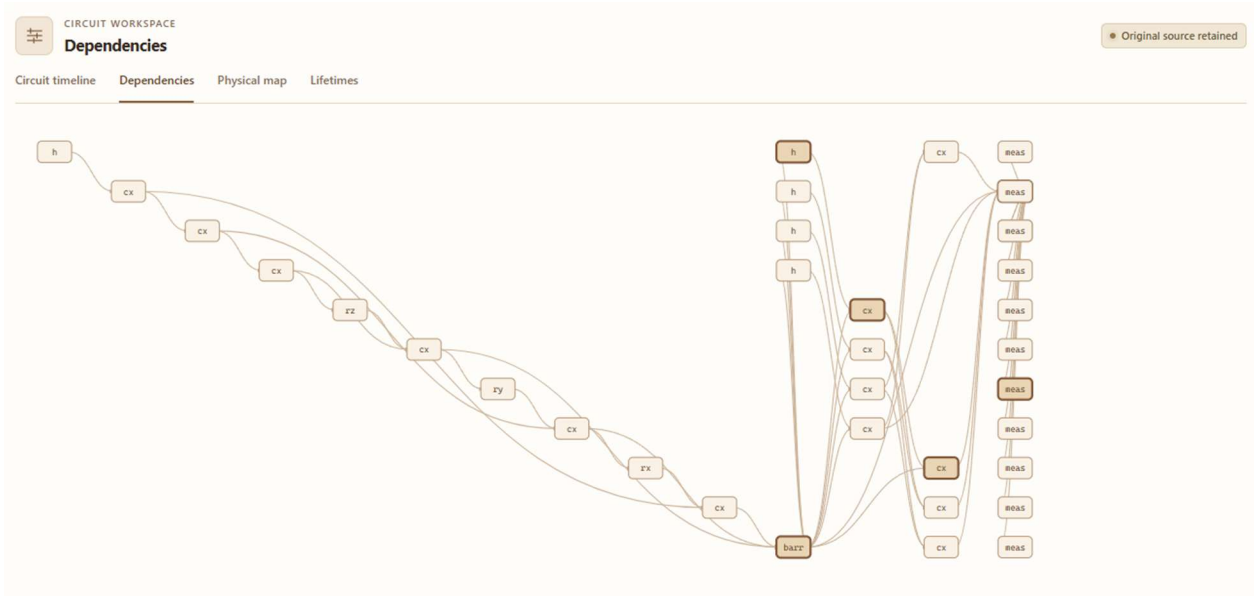


Figure: A screen in Spalax, a custom tool we'll be using in other parts of this course, showing gate dependencies and flow.

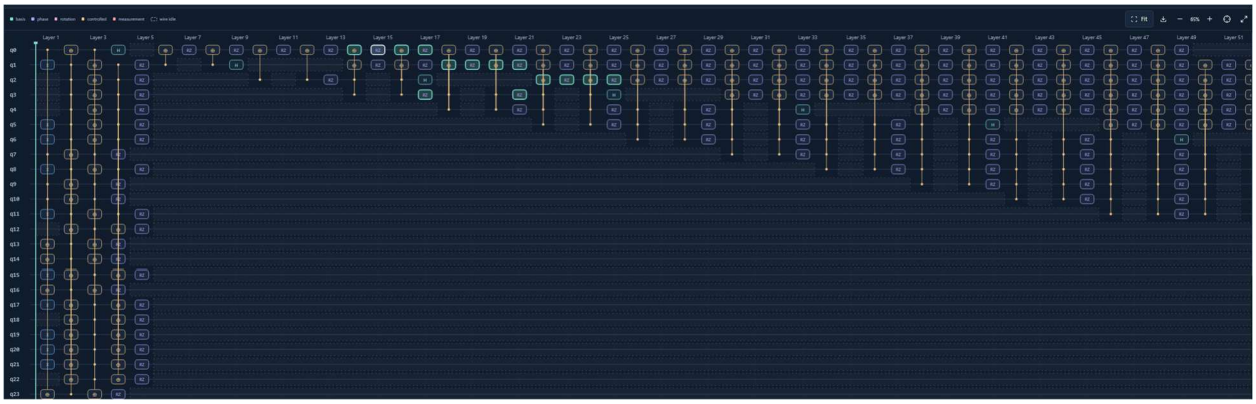


Figure: A screen from Razorwire showing execution layers, a concept that doesn't translate well to classical CPU disassembly, but is important when we look at insertion attacks later.

Visualizing qubit interactions across code

Qubits interact in ways that are radically different from classical bits. These interactions are the basis of quantum code, both legitimate and malicious. In this section, we continue exploring real-world quantum code and go deeper down the rabbit hole as we identify important sections of code by finding patterns in qubit interactions and learning how they work.

- Qubit Interactions
    - Multi-qubit gates
        - CNOT, CZ, SWAP, controlled operations, and how one operation involves multiple qubits
    - Entanglement
        - States that cannot be explained with classical states
    - Quantum interference
        - How amplitudes combine constructively/destructively across computational paths
    - Information propagation
        - How operations spread information through a circuit
    - Interaction topology
        - Logical interaction graph

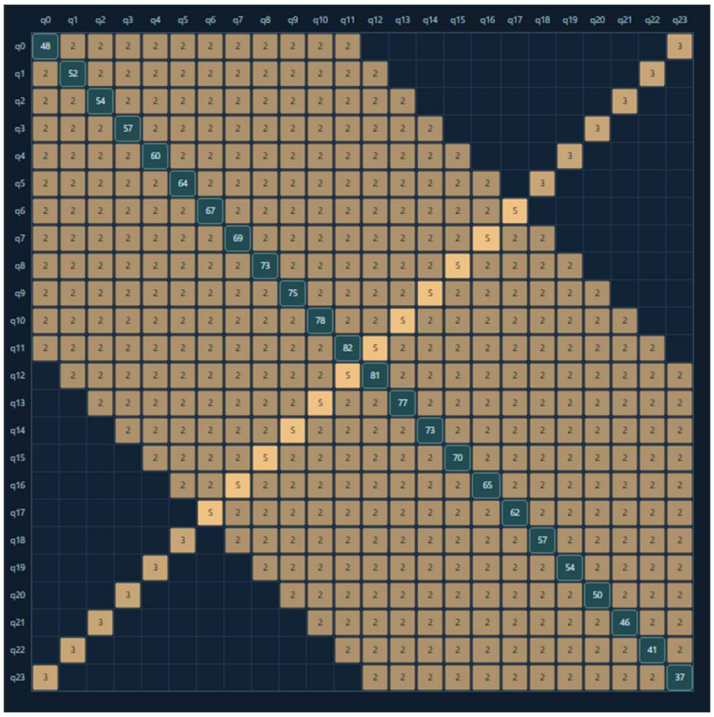


Figure: The "Grid Interaction" view in Razorwire showing the number of interactions between all qubits, one method for identifying sections of code to prioritize reversing.

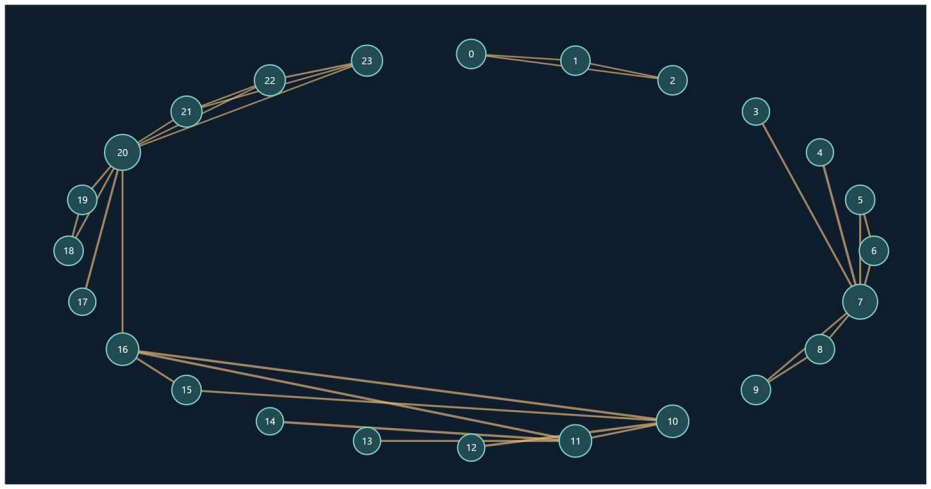


Figure: The "Circle Interaction" map in Razorwire is another visualization that shows clusters of interacting qubits and, conversely, how they are isolated from each other. This signal further divides the input code into the classical notion of "subroutines."

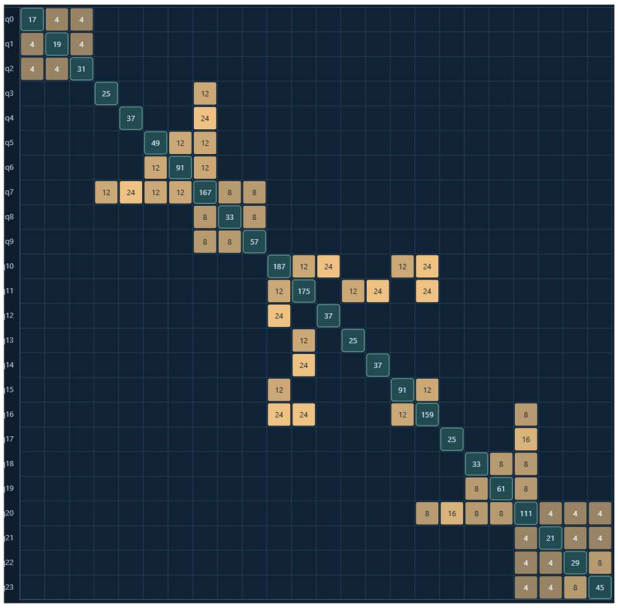


Figure: The "Grid Interaction" view in Razorwire showing the number of interactions between all qubits. Unlike the previous example, this example gives us insights into possible subroutines by identifying reused patterns and which qubits to examine when reversing them.

Identifying interesting qubit operation patterns

The previous lesson taught us how to use the visual tools to identify interesting patterns. This lesson goes lower in the stack by teaching how quantum phenomena are harnessed with gates to implement functionality that is starkly different from classical CPU ASM.

- Adopting CPU disassembly reverse engineering concepts to QC code: Identifying the equivalence of functions, loops, etc.
    - Repeated gate sequences as logical "subroutines"
    - Repeated interaction structures as identification of algorithm registers
    - Repeated layers as identification of algorithm type
    - "Progressive patterns" as identification of information transformation
    - Understanding and identifying "oracle" blocks

Exploring gates and the math behind qubit operations

This lesson is our first *real* dive into qubit states, vectors, and gate matrices. It's our first math-heavy lesson but assumes zero background in math or physics. We will teach the basics gently.

Understanding matrices as a map of input vs output basis

While this lesson continues deeper into the math rabbit hole, we'll teach it differently than it's taught anywhere else. This is not a college class on mathematical theory, but a hands-on class about hacking quantum computers. One trick I've learned for mapping the mathematics describing qubit gates to concrete manipulation of qubit states is to think in terms of "Basis-State Transformation Tables," shown below for a few common gates.

X — Bit flip

Output ↓ / Input →	|0⟩	|1⟩
|0⟩	0	1
|1⟩	1	0

Y — Bit/phase flip

Output ↓ / Input →	|0⟩	|1⟩
|0⟩	0	$-i$
|1⟩	i	0

Z — Phase flip

Output ↓ / Input →	|0⟩	|1⟩
|0⟩	1	0
|1⟩	0	-1

H — Hadamard

Output ↓ / Input →	|0⟩	|1⟩
|0⟩	$1/\sqrt{2}$	$1/\sqrt{2}$
|1⟩	$1/\sqrt{2}$	$-1/\sqrt{2}$

Bell preparation — H then CX

Output ↓ / Input →	|00⟩	|01⟩	|10⟩	|11⟩
|00⟩	$1/\sqrt{2}$	0	$1/\sqrt{2}$	0
|01⟩	0	$1/\sqrt{2}$	0	$1/\sqrt{2}$
|10⟩	0	$1/\sqrt{2}$	0	$-1/\sqrt{2}$
|11⟩	$1/\sqrt{2}$	0	$-1/\sqrt{2}$	0

CCX — Toffoli

Output ↓ / Input →	|000⟩	|001⟩	|010⟩	|011⟩	|100⟩	|101⟩	|110⟩	|111⟩
|000⟩	1	0	0	0	0	0	0	0
|001⟩	0	1	0	0	0	0	0	0
|010⟩	0	0	1	0	0	0	0	0
|011⟩	0	0	0	1	0	0	0	0
|100⟩	0	0	0	0	1	0	0	0
|101⟩	0	0	0	0	0	1	0	0
|110⟩	0	0	0	0	0	0	0	1
|111⟩	0	0	0	0	0	0	1	0

Figures: If you're not yet familiar with how quantum gates work, the tables above will be meaningless. However, if you have a basic understanding of QC gates, these visualizations help beginners bridge abstract mathematics to concrete, real-world effects.

Superposition and Entanglement

Now that we understand the math required to manipulate the quantum phenomena in QPUs, we can use that knowledge to understand two of the primary pillars of quantum logic: superposition and entanglement.

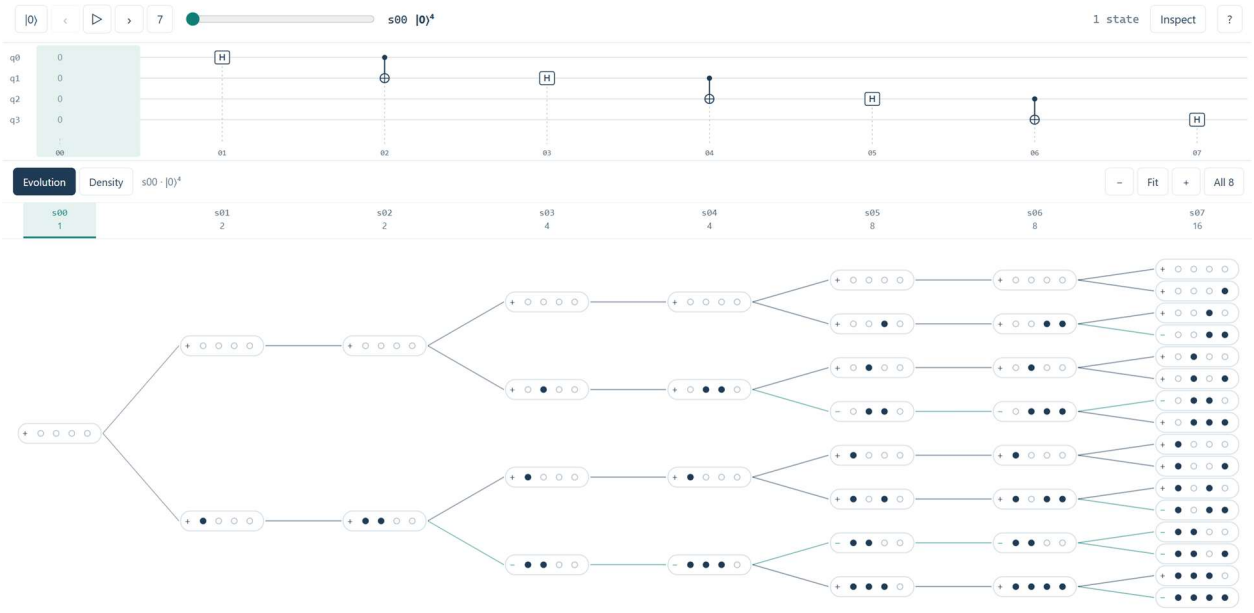


Figure: The lesson in superposition and entanglement is another math-based lesson, but will be taught visually so that attendees will understand the radical capabilities allowing a small number of qubits to represent many different bit values (1 or 0) simultaneously. This graphic shows how 4 qubits in superposition can represent 16 different simultaneous states.

(Technically speaking, it's more correct to say those 4 qubits can represent 16 computational-basis states whose amplitudes describe a joint quantum state, but at times I will use language intended to keep the subject accessible to our intended audience here.)

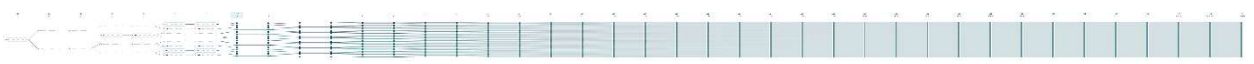


Figure: We will spend time in this section explaining how the number of qubits scales computational probabilities with superposition, and how this is related to quantum computers doing fantastic things – including breaking classical encryption. We will also do a little side quest exploring when different classical encryption paradigms are expected to fall to quantum computation. You can't see it well in this image, but this is like the previous

figure; however, this one shows how only 20 qubits can represent over one million simultaneous states. We'll present it in a way attendees can see and explore themselves.

Phase and Amplitude

This lesson will start with some basic analysis of classical CPU ASM to show how traditional reversing focuses on understanding how instructions manipulate values and control flow. Then, we'll show how phase in qubits does something radically different by changing a qubit's state in ways that are basically unobservable, except under special conditions later. We will start with toy examples to understand the basics of this behavior, then move through simple examples of hiding state, which will be a building block for malicious samples in Day 2. While this is one of the most difficult topics to develop an intuitive understanding of, it's one of the most exciting to learn because of how radically mind-bending it is.

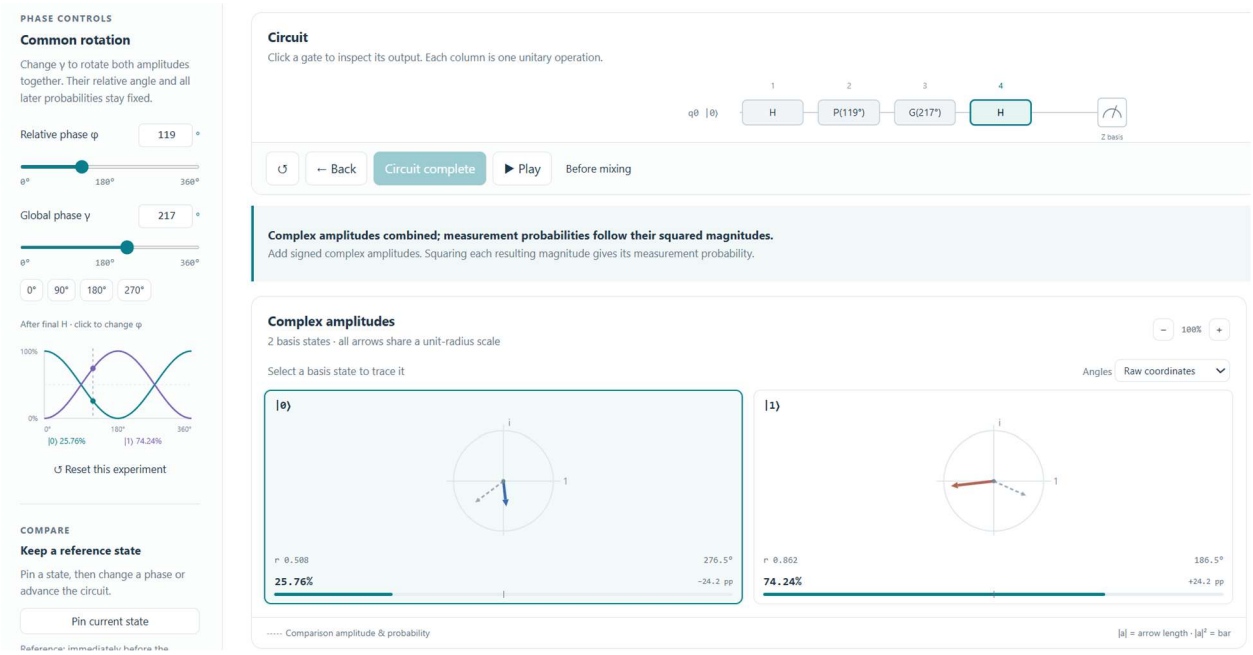


Figure: One of several tools developed for the express purpose of teaching and experimenting with complex QC fundamentals visually. This one is focused on phase.

Workshop: Phase, superposition, and entanglement

These three subjects are so important that we're going to pause here for a mini lab to reinforce both learning and applying these principles.

Day 1 LAB: Reverse engineering an unknown QC program

This semi-guided lab lets attendees combine the techniques and information learned throughout the day into a single workflow to reverse-engineer a real circuit developed by QC researchers. The lab is "semi-guided" in that it will guide them through the steps without telling them exactly what they need to do to solve the questions presented. Clues and hints will be available at the end of the guide. Of course, anyone who would like help with direct answers can receive those from the instructor.



Figure: Additional flow diagrams in Razorwire. These visualizations will feel more natural for users of tools like IDA Pro.

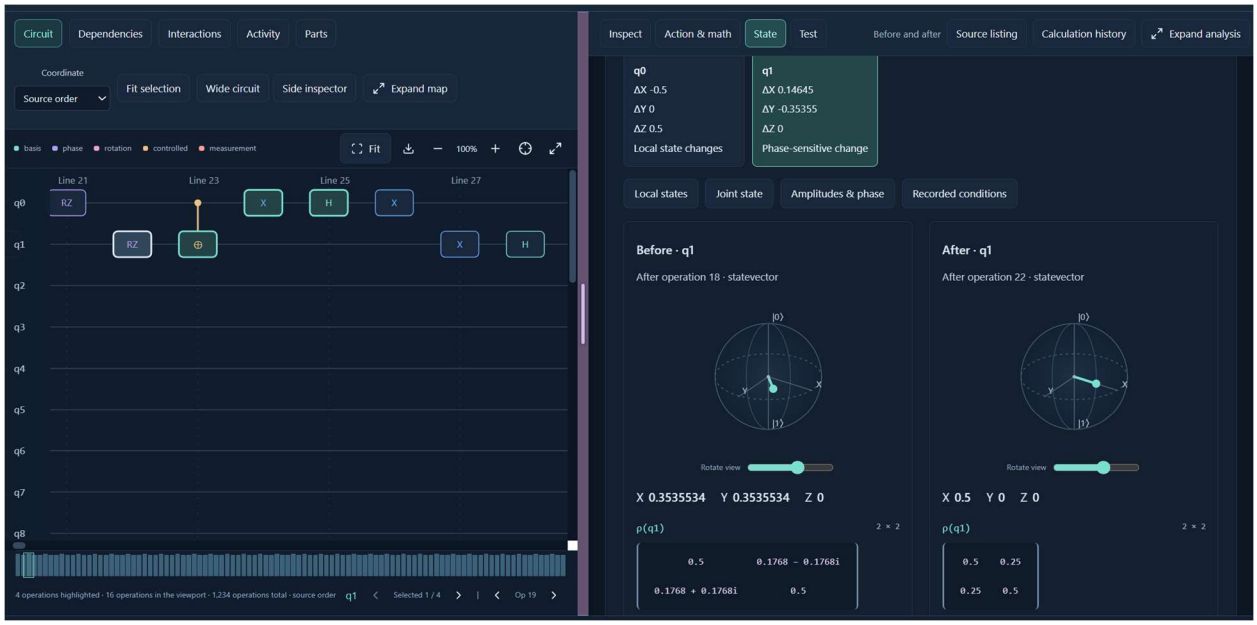


Figure: Additional visualization in Razorwire to assist with reversing the circuit patterns we referred to as “subroutines.”

Day 2: Threats Against Quantum Platforms

Review of important concepts from the previous day

30 Minute review of important concepts from the previous day to reinforce learning through repetition and help warm up our thinking skills at the start of the day.

Day 2 Morning: Learning about the quantum development stack by understanding the threats at each layer

Day 1 helped us appreciate the wildly different world of quantum ASM through the lens of a reverse engineer. We begin Day 2 by understanding where this newfound insight fits in the technology stack, and the spectrum of threats against quantum development workloads.

Classical compute components

Defenders and forensic investigators will likely spend much of their time analyzing the layers of the quantum workload that run on traditional classical computers (CPUs). This section is dedicated to teaching those architectures and the threats/risks at each layer.

Endpoint components

- Application development environments
    - Code execution via circuit deserialization (CVE-2025-2000 and CVE-2026-116499)
    - Memory corruption via malformed circuit state (CVE-2026-116501)
    - Denial of service (CVE-2026-4870, CVE-2026-19795, CVE-2026-116500)
- Transpilers
    - Routing
    - Translation
    - Scheduling
    - Optimization
        - Pre-lesson on the analysis of polymorphic quantum code. *Much more* of this to come in the afternoon.
- Analysis of compiled outputs

        - Overview of papers targeting compiled quantum ML/AI. This space is extremely active. Covers sensitive-data exposure, algorithm reverse engineering, adversarial inputs, and training-data manipulation.

Cloud components

- Cloud compute services
    - Authentication
    - Understanding execution modes (job, session, and batch) through the attacker's lens
    - Queuing
    - Workload routing
    - Job tracking
- Hardware translation
    - Runtime executor
    - Sequence translator
    - Waveform/phase control and readout acquisition
- Overview of compromised clouds and malicious cloud papers

QPU Control Systems

QPU control electronics (quick overview)

- How transpiled quantum instructions are translated into timed control operations.
- Sequencers, reference clock, synchronization

Cryogenic enclosure (quick overview)

- Attenuation, filtering, resonators, amplification, and measurement return paths.
- How physical signal delivery and coupling affect execution fidelity.

QPUs (deeper overview)

- Comparison of major QPU architectures
- Superconducting transmon architecture
    - Analysis of Josephson junctions and virtual quantum particles
        - Is it a particle or is it a wave?
    - Computational states $|0\rangle$ and $|1\rangle$ versus physical higher-energy states $|2\rangle$ and $|3\rangle$
    - Relaxation (T1), dephasing (T2), gate errors, and crosstalk
        - Tie-back to expected QC milestones over the next 5+ years

LAB: Writing a quantum application and examining the artifacts the dev stack creates

In this lab, attendees will develop an application they can run on local simulators and on real quantum hardware in the cloud. We will then examine their endpoint to enumerate the artifacts created on developer workstations when quantum workloads are executed, and assess the investigational value of those artifacts.

LAB: Machine Learning-based approaches to circuit classification (detecting good vs bad circuits)

Students will use a custom-developed Random Forest Classification model, originally inspired by the research here: https://research.ibm.com/publications/algorithmic-intent-leakage-in-transpiled-quantum-circuits-an-initial-reverse-engineering-study, to identify the purpose and features of quantum code. This lab is a building block for later labs, when we will create highly modified malicious circuits to evade detection and identification.

Day 2 Afternoon: Reverse engineering malicious QC code

By the end of this section, attendees will have a broad understanding of the threats QC platforms face, plus a deep technical understanding of several cutting-edge threats and how to reverse-engineer and analyze them.

Threats we'll do technical deep dives into

Hiding data in circuits

- Hidden watermarks
- Results in steganographic research

Circuit hijacking

- Conditional trojans (a primary focus for this course because of their direct relevance to industrial espionage and nation-state economic advantage)

Covert channels

- Higher energy state storage (a fascinating subject with some dedicated focus here, but not one we can get hands-on with in this course)

Spalax: Exploiting opportunity in idle qubits

- This custom-developed tool uses visualizations backed by heuristics and compiler tests to identify “slack space” in circuits – qubits that sit idle long enough to be hijacked for other purposes. The tool allows us to not only identify these opportunities, but also to design “probes” to test and validate the usability of these spaces.
- Spalax allows us to identify possible insertion points for other types of Trojan logic (including the conditional Trojans highlighted previously), as well as helping to verify the complexity of logic that could be inserted within those spaces.

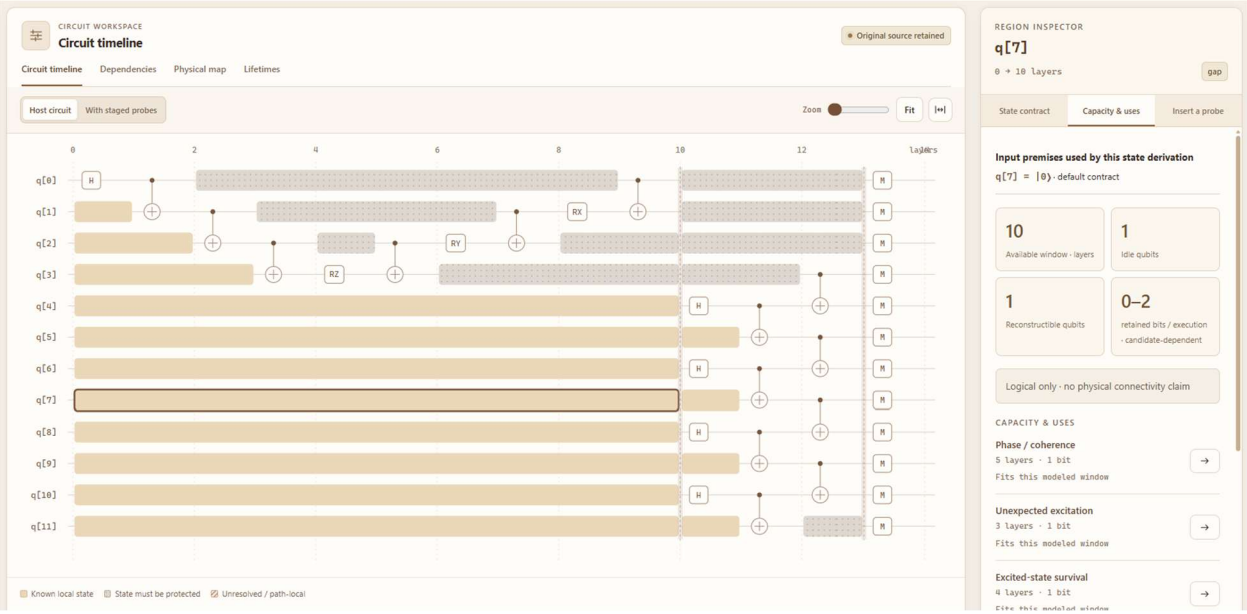


Figure: An interface in the Spalax workflow for identifying “slack space” in circuits and analyzing the ways in which that space can be exploited for other purposes.

A survey of other published threats

- Readout snooping
- Gate/activity interference
- Sequential probes

GateCrasher: Polymorphic insertion and evasion attacks

This has been a primary focus of mine over the past year. Here I will teach attendees how polymorphism works on quantum ASM. We'll further discuss how quantum compilers work, why it is difficult to inject "deceptive noise" into QC code, and how we can insert noise that survives compilation and effectively evades ML detection/classification tooling.

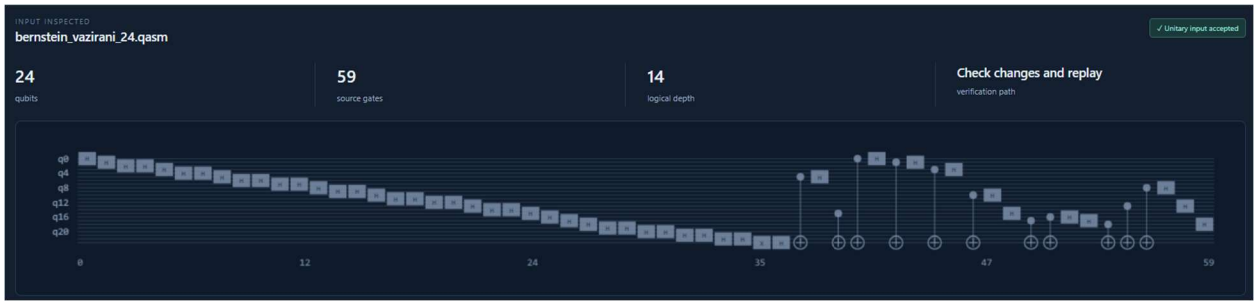


Figure: A simple circuit (Bernstein-Vazirani algorithm on 24 qubits) before it has been modified by GateCrasher.

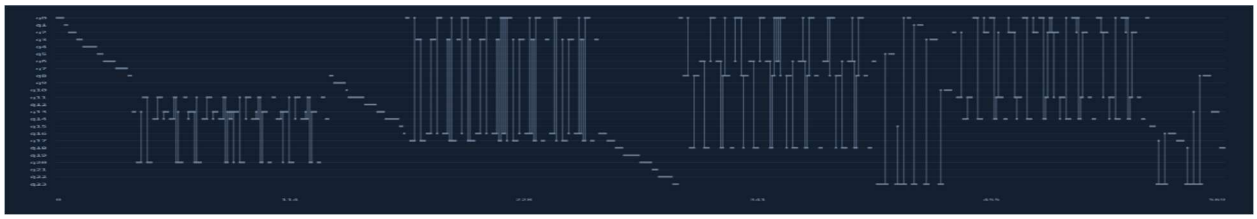


Figure: The exact same circuit. It produces the exact same output as the input circuit. All gates have been recomposed, and complex identity blocks have been inserted, which have survived through transpiler optimization.

LAB: Using GateCrasher to obfuscate a QC trojan and defeat ML detection models

Imo, this is one of the most exciting parts of the training. Here we will work hands-on to embed an operational trojan in a real QC circuit, then modify it to evade detection and reverse engineering.

Day 3 Morning 1.5 Hours - LAB: Reverse engineering an unknown malicious QC circuit (conditional trojan)

This semi-guided lab lets attendees combine the techniques and information learned over the past 2 days into a single workflow to reverse-engineer a malicious circuit modified to avoid identification.

The lab is “semi-guided” in that it will guide them through the steps required to “unpack” the circuit and de-obfuscate it, then determine what it does - without telling them exactly what they need to do to solve the questions presented. Clues and hints will be available at the end of the guide. Of course, anyone who would like help with direct answers can receive those from the instructor.

Day 3: Final Investigation and Reverse Engineering Project

This semi-guided project is designed to both teach and test simultaneously.

In this final project, attendees will assume the role of a research engineer in the CTO’s office of a fictional financial services company. They are developing a quantum algorithm for risk identification. (The first day of the class will give an overview of how enterprises are doing this already.)

Attendees will build quantum code that utilizes a custom-developed “malicious” 3rd-party supply chain package. They will then be guided through identifying artifacts created by this project and answer questions about investigating those artifacts and assessing the threat. They will reverse-engineer the inserted malicious quantum code to determine what effects it had on the enterprise’s research efforts.

The lab is “semi-guided.” We do not assume anyone here is a developer, so we will hand-hold them through environment setup and coding practice. We will help direct them to artifacts and tools for analysis of those artifacts. We will not give them the answers to interpreting those artifacts, but clues and hints will be available at the end of the guide. Of course, anyone who needs help with direct answers can receive those from the instructor. We will ensure all attendees have the knowledge required to pass this evaluation.

Final project scoring

35 points – Reverse engineer the malicious circuit enough to identify its components

35 points – Identify malware keys and effects on the enterprise's research

10 points - Identify execution artifacts

10 points – Identify the compromised package's artifacts

10 points – Report findings and suggest remediations