CyberApocalypse 2026 — Quantum — The Coin That Won't Land

Category: Quantum **Target:** `154.57.164.74:31974` (HTTP / gunicorn + Flask) **Flag:** `HTB{epr_0ath_0p3ns_b0th_w4ys_db1d977f0c6349da2388f9e7fb531e89}`

TL;DR

The service implements a **quantum bit-commitment** ("Oathbinding") scheme and challenges you to survive 32 rounds without ever being caught lying. A commitment that is meant to be simultaneously *binding* (you can't change your vow) and *hiding* is provably impossible in quantum mechanics — the **Mayers–Lo–Chau no-go theorem**. We exploit exactly that: instead of committing to a real bit, we hand the Warden **half of an EPR (Bell) pair** and keep the other half. After the court reveals which basis it will measure in, we measure *our* half in that same basis and steer the sealed qubit to whatever answer we need. Every round holds, and the server confesses the flag.

Recon

The challenge is given only as a socket `154.57.164.74:31974`, and it sends nothing on raw connect. Probing revealed it speaks **HTTP** (gunicorn):

```
$ curl -s http://154.57.164.74:31974/
<title>Oathbinding · The Coin That Won't Land</title> ...
```

`app.js` references an API, and `GET /api/oath` returns the full "writ":

```
{
  "commit": "POST /api/commit {token, slots:[<strand> x strands]}. Each strand is a
             2-qubit circuit over qubits a and b applied to |00>. You keep a. The Warden
             seals b. Returns the court's challenge bit c.",
  "gates":  "single-qubit I X Y Z H S SDG on a or b, e.g. ['H','a']; and CX with
             control,target among a,b, e.g. ['CX','a','b']",
  "open":   "POST /api/open {token, values:[0/1 x strands]} → the Warden measures each b
             in basis c (c=0 → Z, c=1 → X); the oath holds only if every value matches",
  "peek":   "POST /api/peek {token, basis} → measure your kept a-qubits in basis (Z or X),
             returns the outcomes",
  "ritual": "Oathbinding", "rounds": 32, "strands": 8,
  "rule":   "every round must hold or the court walks; open all rounds to be believed"
}
```


Endpoints

Endpoint	Purpose
`POST /api/new`	Open a hearing → `{token, strands:8, rounds:32}`
`POST /api/commit {token, slots}`	Each `slot` is a 2-qubit circuit on `|00>` over qubits `a`, `b`. You keep `a`, Warden seals `b`. Returns `{challenge, round}`
`POST /api/peek {token, basis}`	Measure **your** kept `a`-qubits in `Z` or `X` → `{a_outcomes:[…]}`
`POST /api/open {token, values}`	Warden measures each `b` in the challenge basis `c` (`c=0`→Z, `c=1`→X). Oath holds iff every `value` matches its measurement

---

The vulnerability

A round works like this:

1. **Commit** 8 two-qubit circuits. Qubit `a` stays with us; qubit `b` is "sealed" server-side (but the server keeps the *joint* quantum state).
2. The court reveals a **challenge basis** `c` (Z or X) — *after* the commit.
3. **Open**: for each strand we must predict the outcome of measuring `b` in basis `c`. Any mismatch and "the court walks."

The intended honest behaviour is to commit to a definite value (e.g. put `b` into a Z-eigenstate), which pins you down. The flaw is that the scheme also lets us:

- **choose an arbitrary 2-qubit circuit**, so we can *entangle* `a` and `b` instead of fixing `b`; and
- `peek` **at our own** `a` **qubits after seeing the challenge basis.**

That combination is the textbook attack against quantum bit commitment.

EPR steering

Prepare a Bell pair per strand:

```
['H','a'], ['CX','a','b']   ⇒   |Φ+> = (|00> + |11>)/√2
```

`|Φ+>` has the crucial property of being **perfectly correlated in both the Z and X bases**:

- Z basis: `|Φ+> = (|0>_a|0>_b + |1>_a|1>_b)/√2` → outcomes equal.
- X basis: `|Φ+> = (|+>_a|+>_b + |−>_a|−>_b)/√2` → outcomes equal.

So the plan for each round:

1. Commit a Bell pair on every strand (nothing is decided yet — the coin never lands).
2. Read the challenge basis `c`.
3. **Peek** our `a` qubits in that *same* basis. Measuring `a` collapses each entangled `b` into the matching eigenstate of basis `c`.
4. **Open** with the exact peeked outcomes. When the Warden later measures each `b` in basis `c`, the result is deterministically forced to equal our peek.

Every strand matches, in either basis, so we can satisfy whatever the court throws at us — the commitment is not binding at all.

Exploit

```
import json, urllib.request, urllib.error
B = "http://154.57.164.74:31974"

def post(path, obj=None):
    data = json.dumps(obj).encode() if obj is not None else b""
    req = urllib.request.Request(B+path, data=data,
            headers={"Content-Type":"application/json"}, method="POST")
    return json.loads(urllib.request.urlopen(req).read())

info = post("/api/new")
tok, strands, rounds = info["token"], info["strands"], info["rounds"]
bell = [["H","a"], ["CX","a","b"]]            # |Φ+> Bell pair per strand
slots = [bell]*strands

result = None
for _ in range(rounds):
    c    = post("/api/commit", {"token": tok, "slots": slots})
    basis = "Z" if c["challenge"] == 0 else "X"     # peek in the challenge basis
    vals = post("/api/peek",   {"token": tok, "basis": basis})["a_outcomes"]
    result = post("/api/open", {"token": tok, "values": vals})
    assert result["round_held"]
print("FLAG:", result["flag"])
```

Run:

```
$ python3 solve.py
token: lIn2pTWwl_Zeo04j strands: 8 rounds: 32
...
round 31 ch=0 → {'flag': 'HTB{epr_0ath_0p3ns_b0th_w4ys_...}', 'passes': 32,
                 'round_held': True, 'rounds_done': 32, 'total': 32}
```

Flag

```
HTB{epr_0ath_0p3ns_b0th_w4ys_db1d977f0c6349da2388f9e7fb531e89}
```

The flag itself spells out the lesson: *the EPR oath opens both ways* — a Bell pair lets you answer both the Z and X challenge, so the "box nobody can open" never actually binds you.

Key takeaways

- **Quantum bit commitment cannot be both binding and hiding** (Mayers–Lo–Chau). Any protocol that lets a committer keep a purification/entanglement and defer measurement until after the challenge is broken.
- The concrete cheat is **EPR steering**: commit half of `|Φ+>`, keep the other half, and measure it in the revealed basis to force the correlated outcome.
- Design red flags in the API: the committer chooses an *arbitrary* two-qubit circuit (enabling entanglement of the "kept" and "sealed" qubits) **and** is allowed to measure the kept qubit *after* learning the challenge basis.