Cyber Apocalypse 2026 — Granary Seal

Challenge: Granary Seal
Category: Coding
Target: `154.57.164.74:31719`
Flag: `HTB{th3_0ld_s3qu3nc3_n3v3r_h3s1t4t3s}`

---

1. Challenge Overview

> *The palace granaries are the last thing standing between Crownspire and open starvation. Every ration order that reaches the gatehouse passes through three hands: the clerk who raises it, the countersigner who clears it, and the courier who carries it. Lysa Harrowmere keeps a custody roll for each of the three roles — the hands the gatehouse has actually watched work, entry by entry, over seasons of ordinary orders.*
>
> *Since the Signet shattered, forged orders have been slipping through on habit alone: clean wax, an eager countersign, a courier too calm for a crisis. The fakes are good, but never perfect — a hand that has never touched the roll, a clerk standing where a countersigner belongs, a courier no custody entry has ever named. Lysa needs a count of how many orders in the current batch survive the old sequence: every hand present, every hand where it belongs.*

Stripped of the flavour text, this is a **three-way set-membership check**: three separate whitelists (one per role), a batch of records with one name per role, and a count of records where all three names are on their *own* role's whitelist.

The story deliberately spells out the three distinct failure modes so the intended model is unambiguous:

Story phrase	Failure mode
"*a hand that has never touched the roll*"	name absent from every roll
"*a clerk standing where a countersigner belongs*"	name is known, but on the **wrong** roll for its slot
"*a courier no custody entry has ever named*"	name absent from the courier roll

The middle one is the only interesting hint — it confirms the rolls are **role-scoped, not pooled**. A single flat set of "all known names" would happily wave through a genuine clerk who has been

slotted into the countersigner position. Keeping three independent sets and checking each name against its own set handles this for free, with no extra logic.

2. Recon

The challenge lists a bare `host:port`, but a raw TCP connect returns nothing. An HTTP request reveals a Flask/Werkzeug app:

```
$ curl -s -i --max-time 20 http://154.57.164.74:31719/ | head -6
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.14.0
Date: Sat, 25 Jul 2026 10:52:52 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 28543
Connection: close
```

The page title is just `Coding` — this is HTB's standard **Coding** harness: a Monaco editor, a language selector (`python`, `c`, `cpp`, `rust`), and a *Run Code* button. `/static/js/prompts.js` only holds the generic per-language boilerplate stubs, so the actual problem statement lives in the page markup. Stripping the tags out yields the full specification:

```
$ python3 -c "
import re, html
s = open('index.html').read()
s = re.sub(r'<script.*?</script>', '', s, flags=re.S)
s = re.sub(r'<style.*?</style>',  '', s, flags=re.S)
s = html.unescape(re.sub(r'<[^>]+>', '\n', s))
print('\n'.join(l.strip() for l in s.split('\n') if l.strip()))
"
```

Grepping the inline JavaScript shows a single API route, `/run`:

```
fetch("/run", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ code: code, language: document.getElementById("language-
select").value }),
})
  .then((response) => response.json())
  .then((data) => {
    if (data.challengeCompleted) {
      document.getElementById('flag').innerHTML = data.flag
      showSuccessModal();
    }
```

The server runs the submitted source against hidden test cases and returns the flag in JSON once they all pass — so the whole challenge is scriptable with one `POST`, no browser required.

3. Problem Statement (extracted)

Input

- Line 1: integer `C` — number of clerks on the custody roll.
- Next `C` lines: one clerk name each.
- Next line: integer `CS` — number of countersigners.
- Next `CS` lines: one countersigner name each.
- Next line: integer `R` — number of couriers.
- Next `R` lines: one courier name each.
- Next line: integer `N` — number of orders in the batch.
- Next `N` lines: one order each, as three space-separated names — `clerk countersigner courier`.

Output

A single integer: the number of orders in which every hand appears on its own role's custody roll.

Example

```
Input:
3
aldric.vowmark
bren.irongate
seyna.saltholm
3
voss.ashglass
tal.greywater
mira.crownwall
3
garren.cinders
lysa.stonepass
elric.brinemark
7
aldric.vowmark voss.ashglass garren.cinders
bren.irongate tal.greywater lysa.stonepass
cassian.embervane voss.ashglass garren.cinders
aldric.vowmark forger.oathstone garren.cinders
seyna.saltholm mira.crownwall ghost.saltwind
bren.irongate voss.ashglass elric.brinemark
seyna.saltholm tal.greywater garren.cinders

Expected output:
4
```

Orders 1, 2, 6 and 7 pass. Order 3 fails on the clerk (`cassian.embervane`), order 4 on the countersigner (`forger.oathstone`), order 5 on the courier (`ghost.saltwind`).

4. Approach

The only real trap is the data structure. Names are strings, and the obvious "is this name on the roll?" implementation is a linear scan over a list:

- `name in list_of_clerks` is `O(C)` per lookup.
- With three lookups per order and `N` orders, that is `O(N·(C+CS+R))` — quadratic behaviour that will time out on the hidden tests once the rolls get large.

Swapping each roll to a **hash set** makes every lookup average `O(1)`, giving `O(C + CS + R + N)` overall — linear in the size of the input, which is optimal since every token must be read at least once.

Two smaller details worth getting right:

- **Read the whole stream at once.** `sys.stdin.read().split()` on the entire input and then indexing through the flat token list is dramatically faster than per-line `input()` calls, and it sidesteps trailing-whitespace and line-ending issues entirely. Because every field is a whitespace-free token — names contain a dot but no spaces — a single global `split()` is a lossless parse of the whole file.
- **Duplicate names across rolls are fine.** If someone genuinely serves as both clerk and courier, they appear on both rolls, and the independent sets accept them in either slot. Nothing special is required; the three-set model is already correct.

5. Solution

```
import sys


def main():
    data = sys.stdin.read().split()
    i = 0

    def take_roll():
        nonlocal i
        n = int(data[i])
        i += 1
        roll = set(data[i:i + n])
        i += n
        return roll

    clerks = take_roll()
    countersigners = take_roll()
    couriers = take_roll()

    n = int(data[i])
    i += 1

    valid = 0
    for _ in range(n):
        clerk, countersigner, courier = data[i], data[i + 1], data[i + 2]
        i += 3
        if clerk in clerks and countersigner in countersigners and courier in
couriers:
            valid += 1

    print(valid)


main()
```

The `take_roll()` closure reads a count then slices exactly that many tokens into a set, advancing the shared cursor `i` — the same three-line pattern reused for all three rolls, which keeps the parsing honest about the fixed input layout.

Local verification against the sample:

```
$ python3 solve.py < sample.txt
4
```

6. Submission

POST the source straight to `/run`:

```
import json, urllib.request

code = open('solve.py').read()
req = urllib.request.Request(
    "http://154.57.164.74:31719/run",
    data=json.dumps({"code": code, "language": "python"}).encode(),
    headers={"Content-Type": "application/json"})
print(urllib.request.urlopen(req, timeout=60).read().decode())
```

Response:


```
{"challengeCompleted":true,"flag":"HTB{th3_0ld_s3qu3nc3_n3v3r_h3s1t4t3s}","result":
{}}
```


7. Flag

```
HTB{th3_0ld_s3qu3nc3_n3v3r_h3s1t4t3s}
```

8. Takeaways

- A bare `host:port` in an HTB description is frequently an HTTP service — probe with `curl` before concluding the port is silent.
- The Coding harness is fully driveable from the shell: one `POST /run` with `{code, language}` returns the flag as JSON. Far faster to iterate on than the browser editor.
- The algorithmic content is small but the *modelling* is the point: three role-scoped sets rather than one pooled set. Pooling would silently accept a real clerk placed in the countersigner slot — the exact forgery the story warns about — and it would still pass the provided sample, since none of the seven example orders exercises a cross-role swap. That is a classic hidden-test trap: the sample does not cover the case the prose explicitly calls out.
- Whenever a problem reduces to repeated membership queries, reach for a hash set immediately. Getting the container right is what turns `O(N·M)` into `O(N + M)`, and on these timed judges that is usually the whole difficulty.
- Fittingly for the story: the old sequence doesn't deliberate, it just checks each hand against its own roll and moves on — *the old sequence never hesitates.*