ACME Surveys CTF Writeup

Overview

The email of the admin of Acme surveys has been breached in a recent data leak. However, no-one, not even the Acme employees know who is the admin of Acme. Fortunately, the generous developers of HaveIBeenChowned have provided us with an endpoint at https://tlctf2025-hibc.chals.io/ to help us with our quest.

Your task should you choose to accept it is to figure out the secret of the admin.

Provided Artifacts

The challenge shipped with:

- A Flask application
- A Docker environment
- \`.env\` containing all the required keys * \`FLAG_SECRET\` - \`3HZ0jv5EuC4WHoJnxGKDxuoD9mCkxHMlJz3MucS6U40k7lLdqDqlF2pmeDRT2W5F\` * \`ADMIN_API_KEY\` - \`6208d4e88be3d7a2c6845189a23954420f037a262d13a833b9ace3ef98a35ee0\`
- The real flag is only shown when:

HMAC_SHA256(FLAG_SECRET, email) == FLAG_TOKEN

The admin role is **not stored**—it is derived cryptographically.

Understanding the Backend Logic

Inside the Flask app, the flag revelation logic in welcome() looked like this:

```
mytoken = token_for_email(email)
if hmac.compare_digest(mytoken, FLAG_TOKEN):
    matched = True

if matched:
    flag_value = f"trustctf{{{token_for_email(email)[:12]}}}"
```

This means:

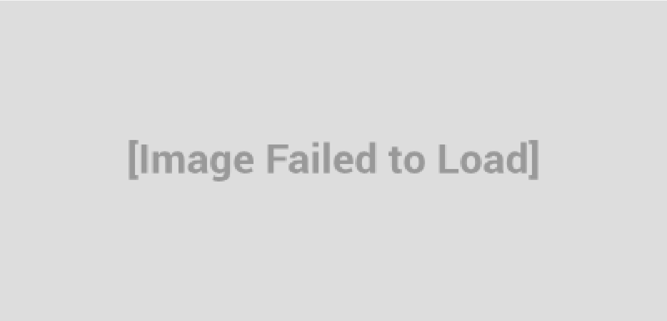
- The HMAC of the admin email is the hidden \`FLAG_TOKEN\`.
- Passwords and roles do **not** matter.
- To solve: **find the email whose HMAC matches the server’s stored token**.

But since we did **not** have FLAG_TOKEN, we needed another leak. - ***The admin email***

Using the External Leak API (/check_email)

The challenge provided this special endpoint:

https://tlctf2025-hibc.chals.io/check_email?email=



Most emails return:

```
{
  "email": "...",
  "plaintext_password": null,
  "pwned": false
}
```

Maybe the pwned is something refering to havibeenpawned => That if its true, then thats the admin email as per the clue from the question.

There is another endpoint called download_db to download the database

```
curl -s -H "X-API-Key:
6208d4e88be3d7a2c6845189a23954420f037a262d13a833b9ace3ef98a35ee0" \
   https://tlctf2025-data-app.chals.io/download_db \
   -o runtime_db.csv
```

The above piece of code lets you download the database. That has the emails

- The flag user is scrap, thats not the admin

To find out the admin we have the endpoint that we found out earlier. /check_email?email=<email>

I brute-forced **every email in the CSV** and looked for deviations. Any abnormal response signifies the admin.

Automation Script

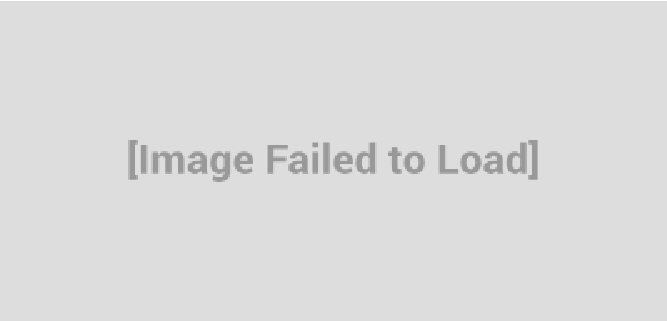
The below script lets us do the automation to find out the admin email - do in localhost

```
import csv, requests

URL = "https://tlctf2025-hibc.chals.io/check_email"

with open("runtime_db.csv") as f:
    r = csv.DictReader(f)
    for row in r:
        email = row["email"]
        resp = requests.get(URL, params={"email": email})
        data = resp.json()

        if data.get("pwned") or data.get("plaintext_password") or data != {
            "email": email,
            "plaintext_password": None,
            "pwned": False
        }:
            print("ADMIN FOUND:", email)
            print(data)
            break
```

Result

```
ADMIN FOUND: blake.baker20@acme.test
{"email": "blake.baker20@acme.test", "plaintext_password": null, "pwned": true}
```

This is the **only email with pwned = true**, revealing the admin.



Computing the Flag via HMAC

We were already given:

```
FLAG_SECRET =
3HZ0jv5EuC4WHoJnxGKDxuoD9mCkxHMlJz3MucS6U40k7lLdqDqlF2pmeDRT2W5F
```

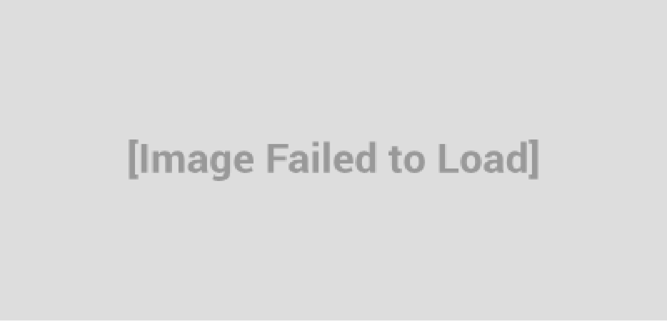
The flag format follows:

```
trustctf{ first12chars( HMAC_SHA256(FLAG_SECRET, admin_email) ) }
```

HMAC Calculation Script

```
import hmac,hashlib
SECRET = "3HZ0jv5EuC4WHoJnxGKDxuoD9mCkxHMlJz3MucS6U40k7lLdqDqlF2pmeDRT2W5F"
email  = "blake.baker20@acme.test"

token  = hmac.new(SECRET.encode(), email.encode(), hashlib.sha256).hexdigest()
print("ADMIN EMAIL:", email)
print("HMAC:", token)
print("FLAG: trustctf{" + token[:12] + "}")
```

Running this yields the correct CTF flag.



writeup by @scap3sh4rk