2/24/2026

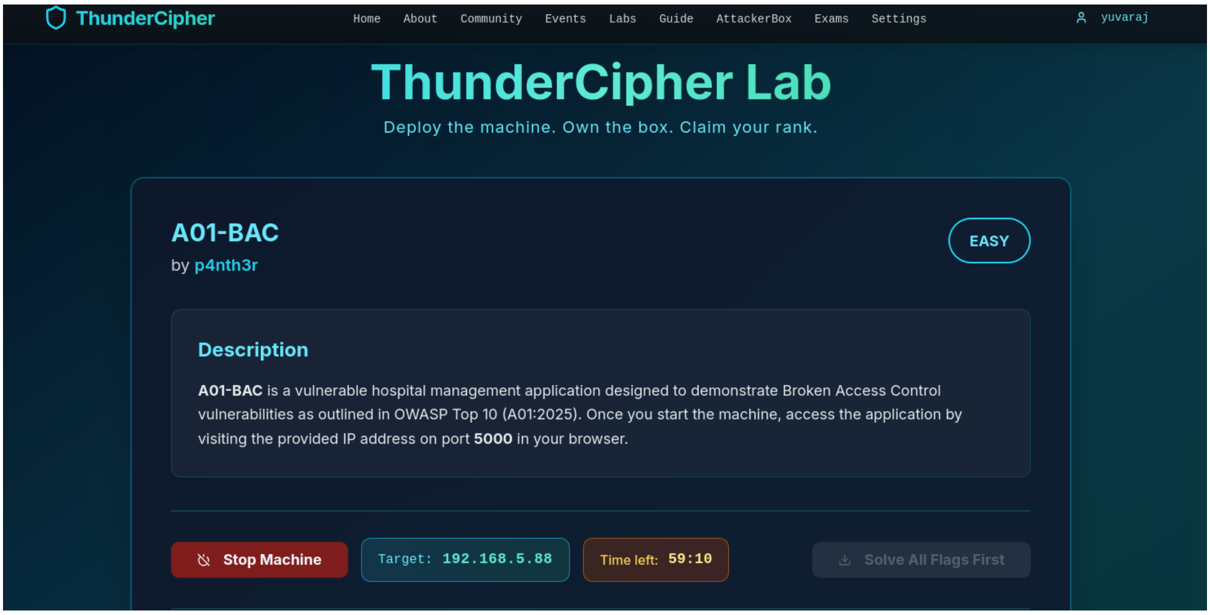
Challenge-02

Solving CTF Labs on ThunderCipher

YUVARAJ M

CityCare Hospital Portal – Writeup

Difficulty: Easy
Category: Web Exploitation
Vulnerability: Broken Access Control (IDOR + Cookie Manipulation)
Target IP: 192.168.5.88
Port: 5000



A01-BAC

by p4nth3r

Description

A01-BAC is a vulnerable hospital management application designed to demonstrate Broken Access Control vulnerabilities as outlined in OWASP Top 10 (A01:2025). Once you start the machine, access the application by visiting the provided IP address on port **5000** in your browser.

Target: 192.168.5.88

Time left: 59:10

Challenge Description

The CityCare Hospital portal is a healthcare management application running on port 5000.

Users can:

- Register
- Login
- View Dashboard
- Access Medical Records
- Attempt to access Admin Panel

The objective was to:

- Gain unauthorized access to the /admin endpoint.
- Access another patient’s medical record and retrieve the hidden flag.

The challenge focuses on analyzing how the application handles authorization and access control.

Enumeration Phase

Step 1 – Port Scanning

Performed an Nmap scan:

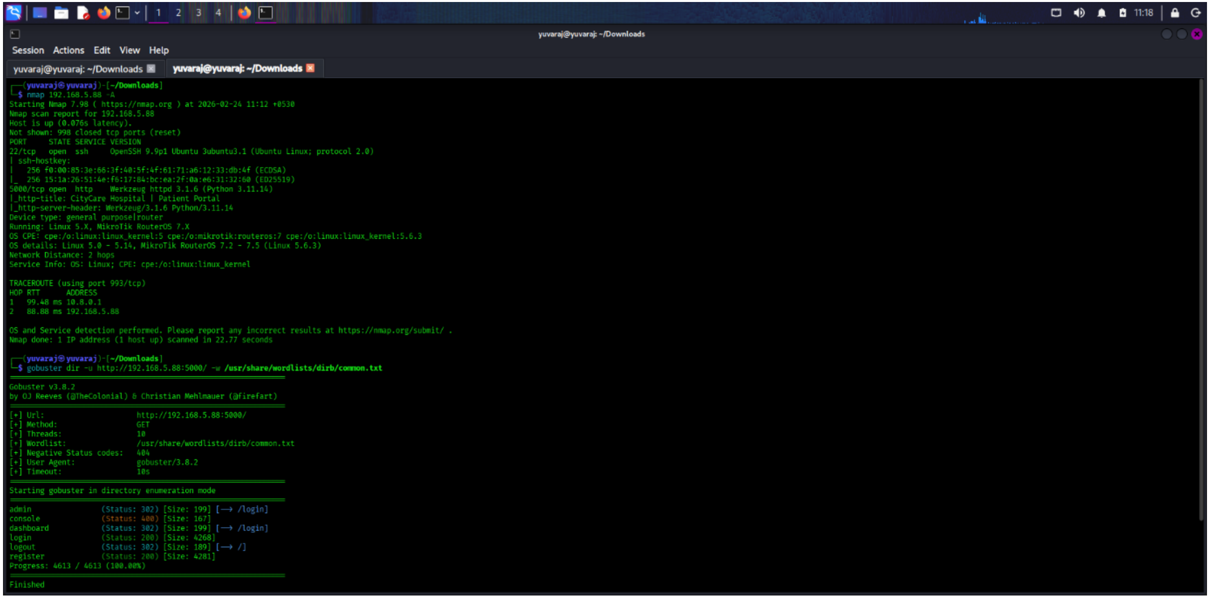
nmap 192.168.5.88 -A

Open ports found:

22 (SSH)

5000 (HTTP – Flask Application)

The web application was running on port 5000.


yuvaraj@yuvaraj: ~/Downloads
Session
Actions
Edit
View
Help
yuvaraj@yuvaraj: ~/Downloads
yuvaraj@yuvaraj: ~/Downloads

```
┌──(yuvaraj㉿yuvaraj)-[~/Downloads]
└─$ nmap 192.168.5.88 -A
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-24 11:12 +0530
Nmap scan report for 192.168.5.88
Host is up (0.076s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.9p1 Ubuntu 3ubuntu3.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 f0:00:85:3e:66:3f:40:5f:4f:61:71:a6:12:33:db:4f (ECDSA)
|_  256 15:1a:26:51:4e:f6:17:84:bc:ea:2f:0a:e6:31:32:60 (ED25519)
5000/tcp open  http    Werkzeug httpd 3.1.6 (Python 3.11.14)
|_http-title: CityCare Hospital | Patient Portal
|_http-server-header: Werkzeug/3.1.6 Python/3.11.14
Device type: general purpose|router
Running: Linux 5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 5.0 - 5.14, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 993/tcp)
HOP RTT      ADDRESS
1   99.48 ms 10.8.0.1
2   88.88 ms 192.168.5.88

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 22.77 seconds

┌──(yuvaraj㉿yuvaraj)-[~/Downloads]
└─$ gobuster dir -u http://192.168.5.88:5000/ -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.8.2
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.5.88:5000/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8.2
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
admin                (Status: 302) [Size: 199] [--> /login]
console              (Status: 400) [Size: 167]
dashboard            (Status: 302) [Size: 199] [--> /login]
login                (Status: 200) [Size: 4268]
logout               (Status: 302) [Size: 189] [--> /]
register             (Status: 200) [Size: 4281]
Progress: 4613 / 4613 (100.00%)
===============================================================
Finished
```

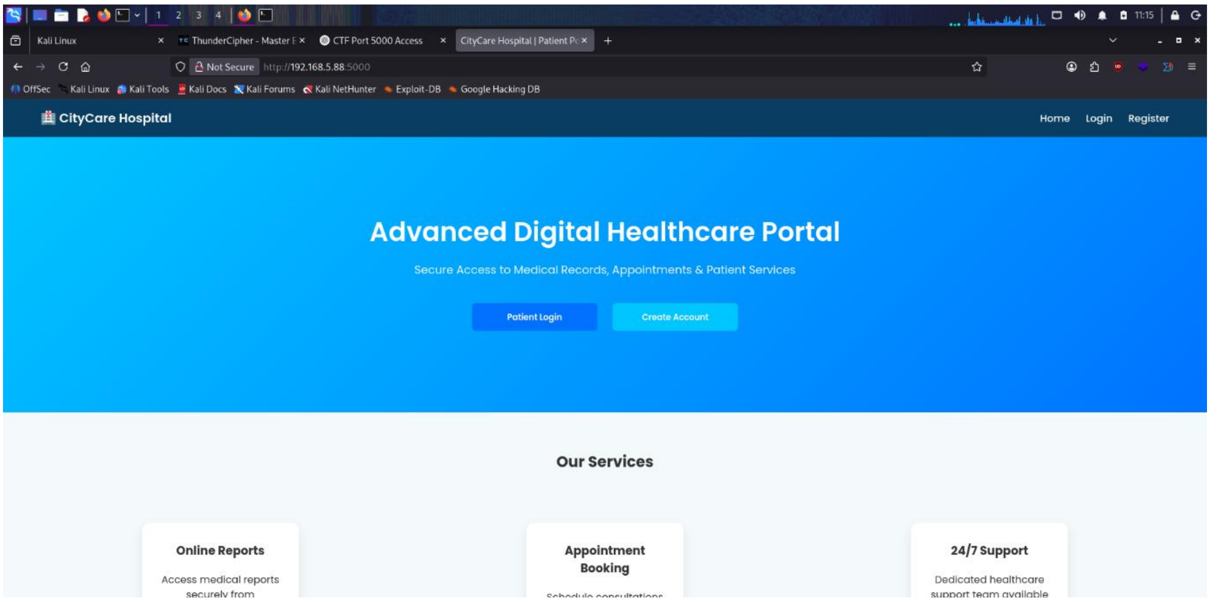
Accessed:

http://192.168.5.88:5000
Application Analysis

Step 2 – Exploring the Application

After registering and logging in, the application provided:

- Dashboard
- Medical Record section
- Admin tab (restricted)



Advanced Digital Healthcare Portal

Secure Access to Medical Records, Appointments & Patient Services

Our Services

Online Reports

Access medical reports securely from

Appointment Booking

Schedule consultations

24/7 Support

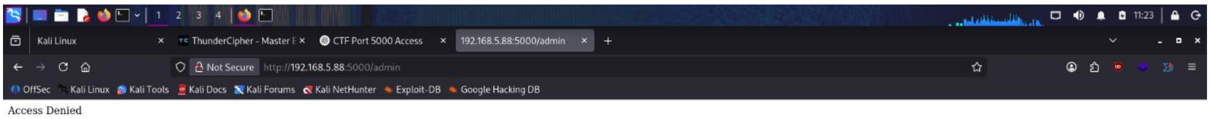
Dedicated healthcare support team available

When accessing:

http://192.168.5.88:5000/admin

The response was:

Access Denied



Access Denied

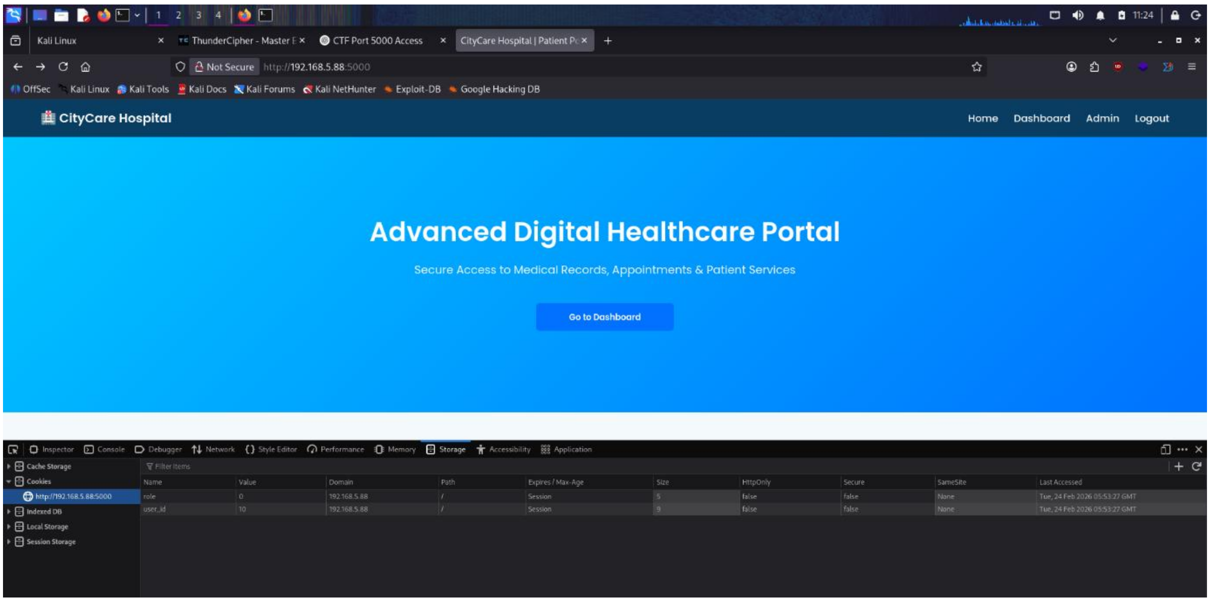
This indicated role-based access control was implemented.

FLAG 1 – Admin Panel Access

Step 3 – Inspecting Cookies

Opened Firefox Developer Tools:

F12 → Application → Cookies



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
role	0	192.168.5.88	/	Session	5	false	false	None	Tue, 24 Feb 2026 05:53:27 GMT
user_id	10	192.168.5.88	/	Session	9	false	false	None	Tue, 24 Feb 2026 05:53:27 GMT

Observed cookies:

- role = 0
- user_id = 10

Observations:

role = 0 → **Normal user**

user_id = 10 → **Logged-in user**

- Cookies were not HttpOnly
- Cookies were client-controlled

This suggested the application was relying on client-side cookies for authorization.

Step 4 – Privilege Escalation via Cookie Manipulation

Modified cookie value:

Original:

role = 0

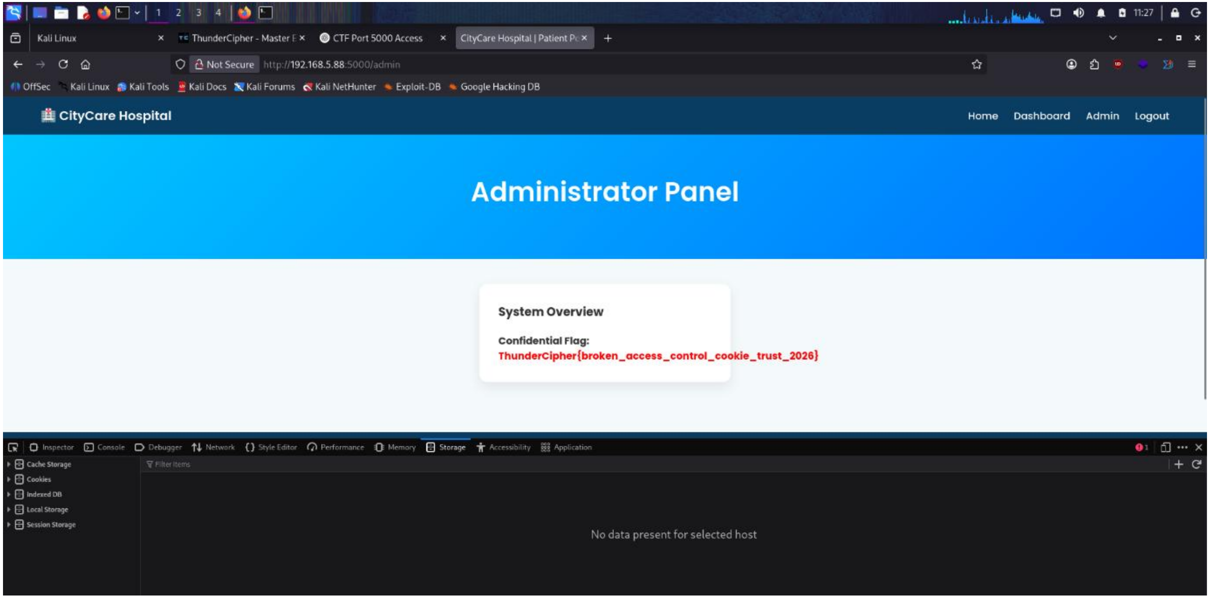
Modified:

role = 1

After changing the role cookie, refreshed:

http://192.168.5.88:5000/admin

The **Admin panel** loaded successfully.



Administrator Panel

System Overview

Confidential Flag:
ThunderCipher{broken_access_control_cookie_trust_2026}

No data present for selected host

Step 5 – Flag Captured

The admin page displayed:

ThunderCipher{broken_access_control_cookie_trust_2026}

Vulnerability Explanation – Vertical Privilege Escalation

The application trusted the role value stored in the client-side cookie.

Example vulnerable logic:

```
if request.cookies.get("role") == "1":
    allow_admin()
```

Since cookies are fully controlled by the client, modifying the value resulted in unauthorized admin access.

This vulnerability is categorized under:

- **Broken Access Control**
- **Vertical Privilege Escalation**
- **OWASP Top 10 – A01**

FLAG 2 – Accessing Another Patient Record (IDOR)

Step 6 – Analyzing Medical Record Endpoint

While viewing medical records, the following URL was observed:

/medical_record?id=10

The id parameter controlled which patient record was displayed.

This indicated a **potential IDOR vulnerability**.

Step 7 – Exploiting IDOR

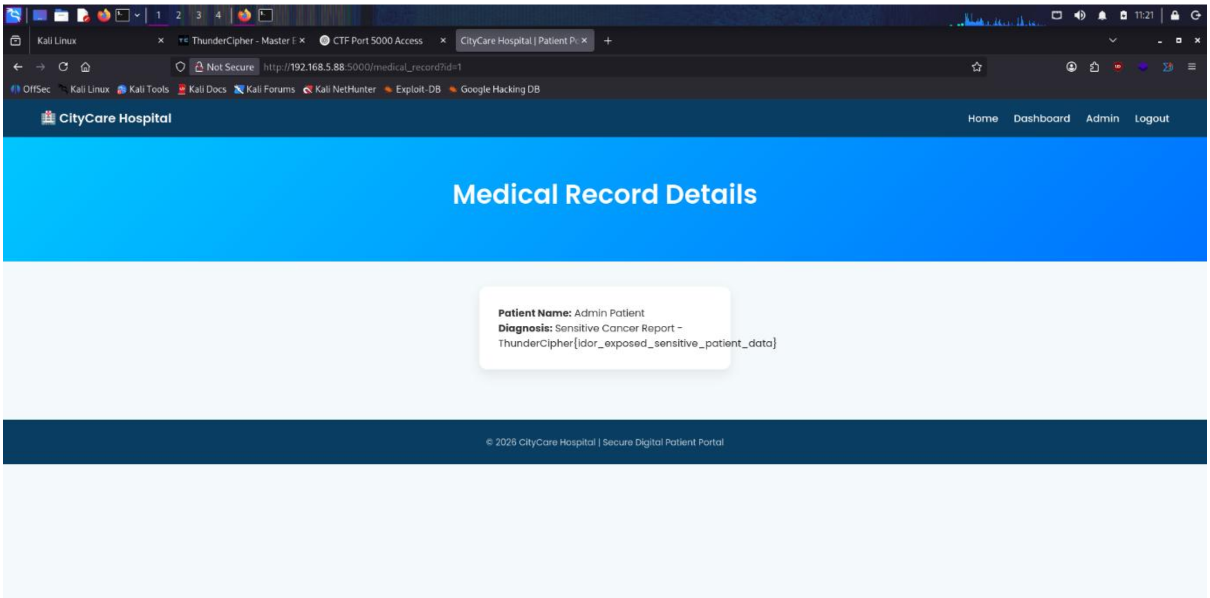
Modified the URL:

Original:

/medical_record?id=10

Modified:

/medical_record?id=1



http://192.168.5.88:5000/medical_record?id=1

Medical Record Details

Patient Name: Admin Patient
Diagnosis: Sensitive Cancer Report - ThunderCipher{idor_exposed_sensitive_patient_data}

© 2026 CityCare Hospital | Secure Digital Patient Portal

The application displayed:

Patient Name: Admin Patient
Diagnosis: Sensitive Cancer Report - **ThunderCipher{idor_exposed_sensitive_patient_data}**

Step 8 – Flag Captured

Second flag obtained:

ThunderCipher{idor_exposed_sensitive_patient_data}

Vulnerability Explanation – IDOR

The server directly used the id parameter to fetch medical records without verifying ownership.

Example vulnerable logic:

```
record_id = request.args.get("id")
return get_medical_record(record_id)
```

Missing validation:

```
if record.owner_id != session["user_id"]:
    deny_access()
```

This allowed any authenticated user to access other patient’s records.

This vulnerability is categorized under:

- **Insecure Direct Object Reference (IDOR)**
- **Horizontal Privilege Escalation**
- **Broken Access Control**

Impact

If deployed in a real-world healthcare system, this vulnerability could allow:

- Unauthorized access to patient medical data
- Exposure of confidential records
- Unauthorized administrative access
- Data manipulation or deletion
- Regulatory violations (HIPAA / GDPR)

Severity: Critical

Mitigation

1 – Do Not Trust Client-Side Role

Authorization should be enforced server-side:

```
if session["role"] != "admin":
    return "Unauthorized"
```

2 – Validate Resource Ownership

Before serving records:

```
if session["user_id"] != record.owner_id:
  return "Forbidden"
```

3 – Secure Cookies

- Use HttpOnly
- Use Secure flag
- Use SameSite
- Store sessions server-side

Never trust client-controlled identifiers for authorization.

Lessons Learned

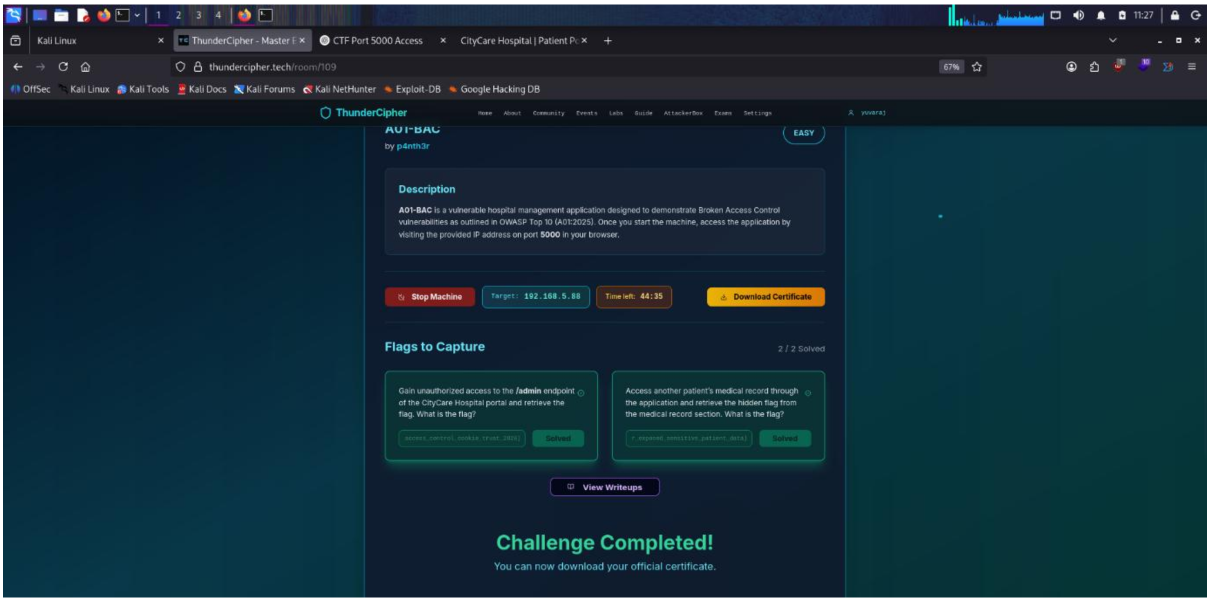
- Always inspect cookies for role-based logic
- If you see IDs in URLs → test for IDOR
- Client-side authorization is insecure
- Broken Access Control is one of the most critical web vulnerabilities
- Small logic flaws can lead to complete system compromise

Attack Flow Summary

- Scanned target → Found port 5000
- Registered test account
- Attempted to access /admin → Access Denied
- Inspected cookies → Found role=0
- Modified role=1 → Gained admin access
- Retrieved first flag
- Inspected medical_record endpoint
- Modified id parameter
- Accessed admin patient record
- Retrieved second flag

Final Flags

ThunderCipher{broken_access_control_cookie_trust_2026}
ThunderCipher{idor_exposed_sensitive_patient_data}



A01-BAC

by p4nth3r

EASY

Description

A01-BAC is a vulnerable hospital management application designed to demonstrate Broken Access Control vulnerabilities as outlined in OWASP Top 10 (A01:2025). Once you start the machine, access the application by visiting the provided IP address on port **5000** in your browser.

Target: 192.168.5.88

Time left: 44:35

Flags to Capture

2 / 2 Solved

Gain unauthorized access to the **/admin** endpoint of the CityCare Hospital portal and retrieve the flag. What is the flag?

access_control_cookie_trust_2026}

Solved

Access another patient's medical record through the application and retrieve the hidden flag from the medical record section. What is the flag?

r_exposed_sensitive_patient_data}

Solved

Challenge Completed!

You can now download your official certificate.