Insecure Direct Object Reference (IDOR)

- **Student:** Marlon Brenes
- **Program:** Master's in Cybersecurity
- **Course:** INCS-745 – IDHE (NYIT Vancouver)
- **Professor:** Dr. Sara Khanchi

Table of Contents

Definition:
IDOR occurs when attackers manipulate object identifiers (e.g., URLs, parameters) to access/modify unauthorized data.

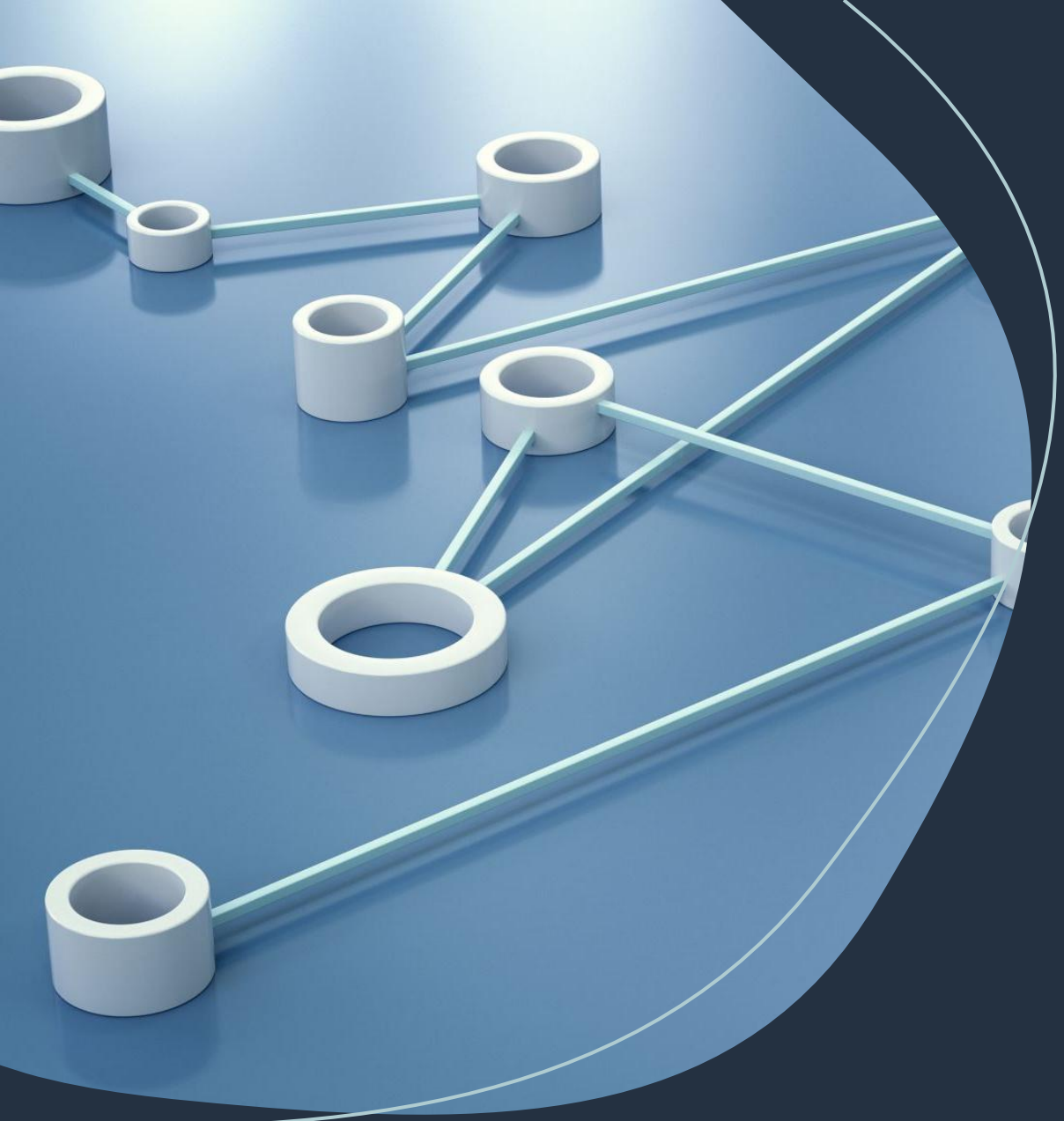
- **Root Cause**: Missing access control checks.
- **Risk**: Unauthorized data exposure, tampering, or deletion.

Scenario: User profile access via URL:

https://example.org/users/123

- **Vulnerability**: Changing 123 to 124 exposes another user's data.

- 123 and 124 are GUIDs or UUIDs (Users identifiers).



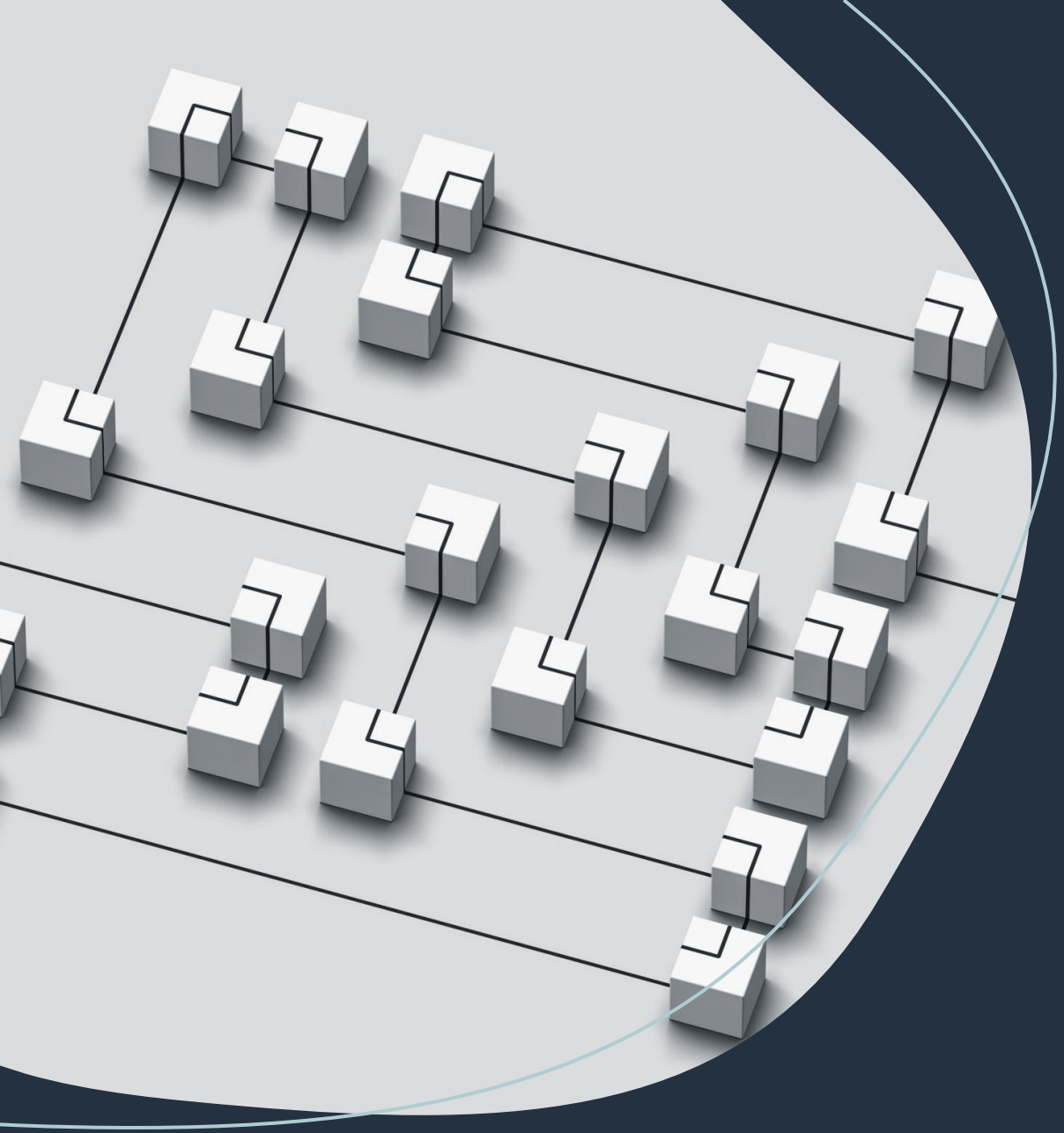
Identifier Complexity – A False Sense of Security

- **Complex IDs** (GUIDs/UUIDs):
  - Make guessing harder but **do not replace access control**.
- **Key Point**: Attackers can still exploit leaked/complex IDs without proper checks.

Mitigation 1 – Access Control Checks

- **Mandatory Step**: Verify user permissions **every time** an object is accessed.
- **Framework Integration**: Use built-in methods (e.g., Ruby on Rails):

```
# Vulnerable: Searches all projects
@project = Project.find(params[:id])  (# Please Don't!)
# Secure: Filters by current user
@project = @current_user.projects.find(params[:id])
```



Mitigation 2 – Avoid Exposing Identifiers

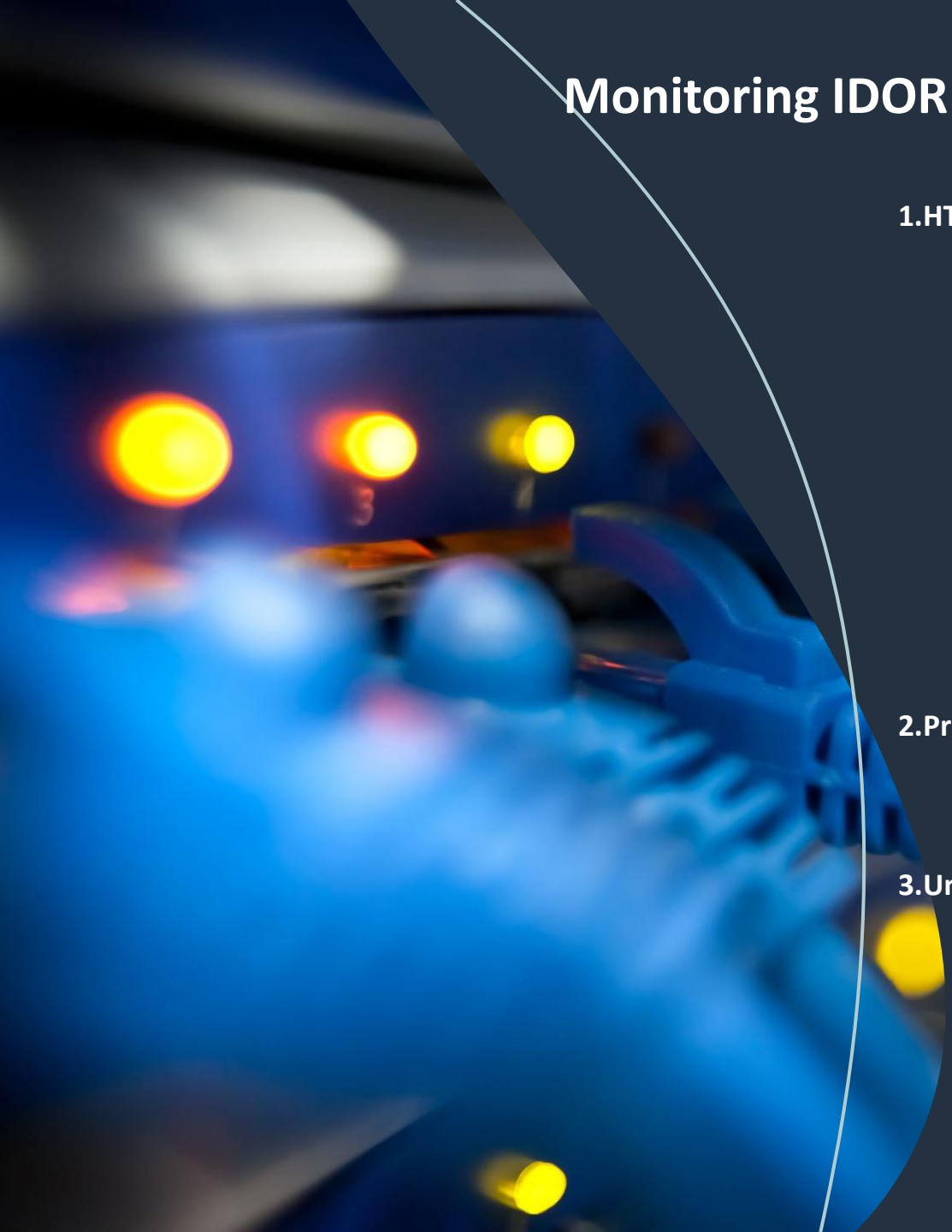
- **Best Practices**:
  - Derive user identity from **session data**, not URLs/POST bodies.
  - In multi-step flows, store IDs in server-side sessions.

Summary & Key Takeaways

1. **Enforce access control** for every object access.
2. **Avoid exposing identifiers** in URLs/forms.
3. Use **UUIDs** as a defense-in-depth layer.
4. Leverage **framework-specific security features**.

Tools for Exploiting IDOR Vulnerabilities

- Burp Suite
- OWASP ZAP (Zed Attack Proxy)
- Postman
- Browser Developer Tools
- Many more...



Monitoring IDOR with Wireshark – Build a good Telemetry.

1.HTTP Requests with Exposed Identifiers

- Filter: http.request.method == "GET" or http.request.method == "POST".
- Look For:
  - URLs with numeric/sequential IDs (e.g., /api/users/123, /download?file_id=456).
  - POST parameters like user_id=789 or account_number=9876.

2.Predictable Patterns in Identifiers

- Example: Sequential numbers (e.g., 1001 → 1002 → 1003) or easily guessable strings.

3.Unauthorized Access Responses

- Filter: http.response.code == 200 for successful unauthorized access.
- Compare: Legitimate user requests vs. tampered requests (e.g., user 123 accessing user 124's data).

4.Missing Access Control Tokens

- Look For: Requests lacking session tokens, JWTs, or cookies to validate permissions.
- Filter: http.cookie or http.authorization.

5.Sensitive Data in Responses

- Filter: http contains "password" or http contains "SSN".
- Example: A response containing another user's private data after ID tampering.

References

- **OWASP IDOR Prevention Cheat Sheet**: [Link]
- **Additional Resources**:
  - OWASP Top 10
  - Web Framework Security Guides (e.g., Django, Rails).

Thank You!