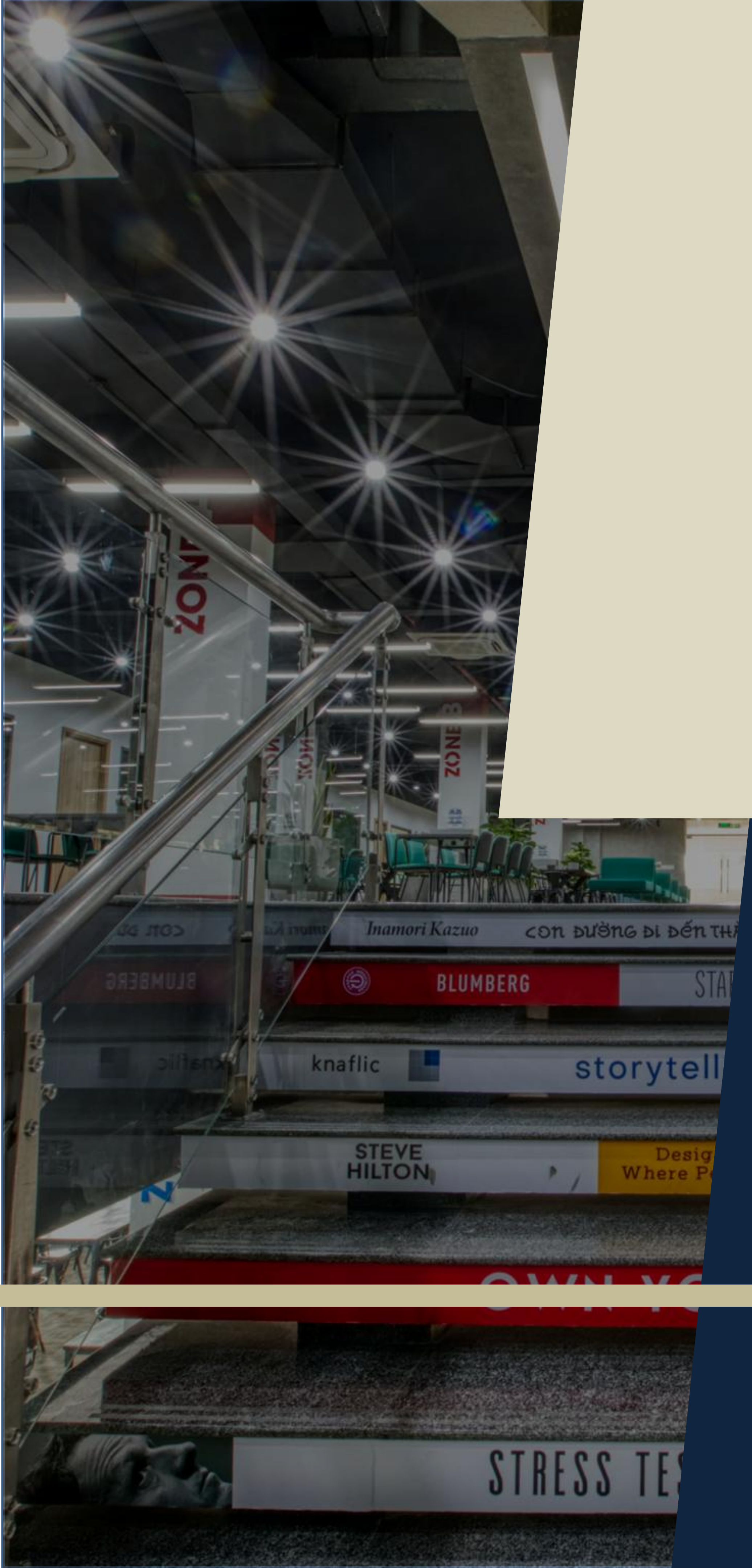


TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Backend Development

Chapter 7: Security

TABLE CONTENT

Why is security important?

- Backend typically handles: personal, financial, access data
- A small vulnerability can lead to:
  - Data leakage
  - Loss of credibility
  - Financial/Legal Damages

Lesson objectives

- Identify common vulnerabilities in the backend
- Applying practical security rules and patterns
- Introducing the protection tools in ASP.NET Core

COMMON SECURITY VULNERABILITIES

Common Vulnerabilities Overview

- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Broken Authentication
- Insecure Deserialization
- Security Misconfiguration

Common Vulnerabilities Overview

SQL Injection

Hacker inserts SQL code into input

Example:

```
SELECT * FROM Users WHERE Username = 'admin' OR '1'='1'
```

Solution:

Use ORM (Entity Framework)

Use parameterized queries

Common Vulnerabilities Overview

Cross-Site Scripting (XSS)

Hackers insert scripts into forms, URLs

- If the render frontend does not escape HTML→ Execute malicious code

Solution:

- Escape output
- CSP (Content Security Policy)

Common Vulnerabilities Overview

Cross-Site Request Forgery (CSRF)

Logged-in users→ hacker send Fake request
Solution:
Use AntiForgeryToken in ASP.NET
Check Origin/Referer

Common Vulnerabilities Overview

Broken Authentication

Common errors:

- Save plain-text passwords
- JWT without expiration
- No control session/token revocation

Solution:

- Use the Identity framework or Oauth
- Hash + salt password (PBKDF2, bcrypt...)

EIU

Common Vulnerabilities Overview

Insecure Deserialization

Get input (JSON/XML/binary) and deserialize uncontrolled

- Easily lead to malicious code execution

Solution:

- Block unclear object bindings
- Checking Input → whitelist class/fields

Common Vulnerabilities Overview

Security Misconfiguration

Configuration errors are the leading causes:

- Debug mode enabled on production
- Security header not available (HSTS, X-Frame-Options)
- Use the default admin account

Solution:

- Automatically check config when deploying
- Use of linter, scanner

BEST PRACTICES

Best Practices

Secure encryption and storage

- Password: hash (bcrypt, PBKDF2)
- JWT/Token: should have exp, signed with a strong key
- Sensitive data: AES encryption at rest

Best Practices

Roles management and decentralization

Principle of Least Privilege:

- Users can only do what is necessary

ASP.NET Core: use Role-based, Policy-based Authorization

- Add [Authorize(Roles = "Admin")]

Best Practices

API Protection

- Rate Limiting
- Validate input with ModelState
- Use HTTPS (required)
- Check token/claims trong middleware

Best Practices

Logging and Monitoring

- Log request/response details(except for passwords)
- Detect unusual behavior(Wrong login multiple times)
- ASP.NET Core: dùng ILogger, Serilog, Application Insights

Best Practices

Testing & Auditing

- Pentest periodically
- Use the tools:
  - OWASP ZAP
  - Dotnet Security Analyzer
- Save trace requests for auditing if something goes wrong

Best Practices

Configuration security and secrets

- Don't commit the connection chain or API key
- Using User-Secrets in dev
- Using Azure Key Vault, AWS Secrets Manager in production

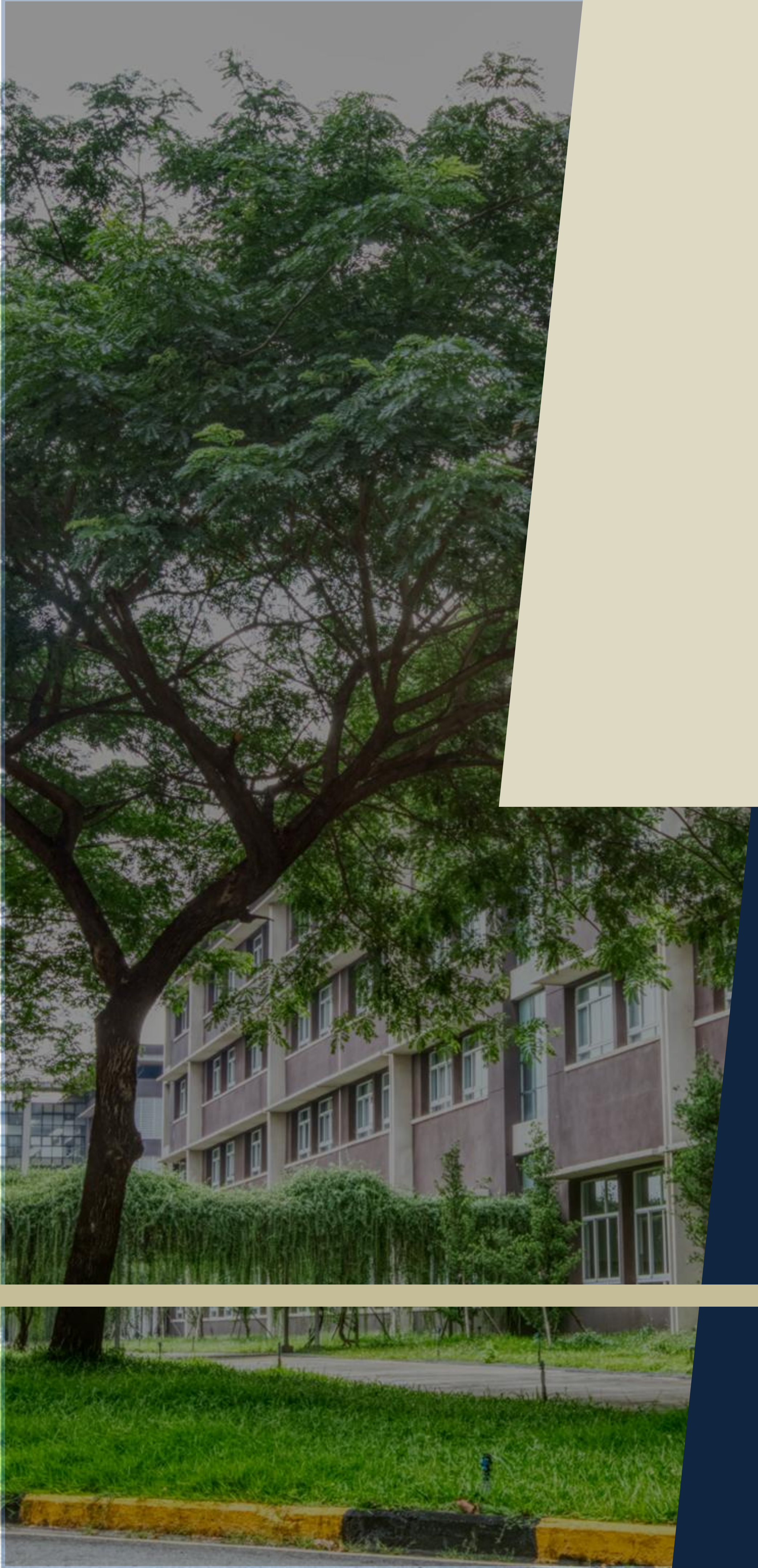
Best Practices

Automate security audits

Integration into CI/CD:

- Static Code Analysis (SonarQube, Snyk
- Check package vulnerability(NuGet Audit)

GitHub Actions to run automated security scans



TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Start your future at EIU

Thank You