# Written Assignment 15%

Complete the assignment and submit by end of week 12 (By 7 Jan 2024, Sunday, 23:59) Submission is via LMS under week 12 section.

## SECTION A

**Purpose**

To investigate good security features offered by a typical web application.

## Tasks

Describe the good security features offered by a typical web application based on the features listed below. Use the learning material from NYP LMS (Week 1 to 9) as well as some web research.

* **Pick** 2 Web App features from SECTION B that you will implement.
* **List** and **describe** 3 security features that you will implement for each of the “Web Application Features” that you picked. A good example of a security feature for login page feature will be “account lockout”.
* For each of the security feature:
  + Explain how the security feature should be implemented. E.g. describe the steps or process involved.
  + Explain the security issue(s) or problem that you are trying to avoid or mitigate. You may quote references from OWASP or SANS.
  + Provide reasons to support your solution(s) or implementation. (e.g. explain why do you think your solution will work?)
* You are not required to code the application. However, you may include code snippets or screenshots to help illustrate your solution.

## Assumption/rules:

* All the web application is assumed to be a typical ASP.NET or ASP.NET Core.
* No limit to the number of words for this assignment.
* You may provide screenshots to illustrate your solution(s)

## SECTION B

**Web Application Features (Choose any 2 from the list) – 15 marks each**

1. Login page
2. Forget Password or account recovery process
3. Payment page with Credit card Info (Credit card info is required to be saved in Database)
4. Account Registration Page
5. Data encryption and archival

Report Template (ARIAL, FONT 12)

Student Full Name : Lee Jaryl

Admin No : 222441U

Tutorial Group : IT2163-02

I have chosen the following security features for my assignment:

* 1. Login Page (Page 3-5)
  2. Data Encryption and Archival (Page 6-8)

Statement on Plagiarism

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication.

I also certify that this assignment/report has not previously been submitted for assessment in any other unit, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons.

|  |  |
| --- | --- |
| Signature |  |
| Date | 7/1/2024 |

# 1. Login Page

**Assumptions:**

* The web application is built using ASP.NET or ASP.NET Core.
* User authentication is based on a username and password.
  1. **Weak Password Policy**

Weak password policies can lead to vulnerabilities, making it easier for attackers to gain unauthorized access.

**References:**

o OWASP: A3:2017 - Sensitive Data Exposure

o SANS: CWE-521: Weak Password Requirements

**Mitigation Techniques:**

Implement a strong password policy that includes requirements such as a minimum length, a mix of uppercase and lowercase characters, numbers, and special symbols.

**Implementation:**

* Update the authentication configuration to enforce the desired password policy. Here is an example in ASP.NET Core:

*A screen shot of a computer program

Description automatically generated*

This ensures that passwords must have a mix of characters, digits, and symbols, meeting the specified length.

* 1. **Brute Force Attacks**

Unrestricted login attempts can make the application vulnerable to brute force attacks.

**References:**

* OWASP: A6:2017 - Security Misconfiguration
* SANS: CWE-307: Improper Restriction of Excessive Authentication Attempts

**Mitigation Techniques:**

Implement account lockout after a specified number of unsuccessful login attempts to prevent brute force attacks.

**Implementation:**

* Configure account lockout settings in ASP.NET Core:

A screen shot of a computer program

Description automatically generated

This code sets the maximum number of failed login attempts and the lockout duration.

* 1. **Insecure Session Management**

Inadequate session management can expose users to session hijacking or fixation attacks.

**References:**

* OWASP: A3:2017 - Sensitive Data Exposure
* SANS: CWE- 384: Session Fixation

**Mitigation Techniques:**

Implement secure session management practices, such as using secure cookies and regenerating session IDs after a successful login.

**Implementation:**

* Configure session options in ASP.NET Core:

A screen shot of a computer program

Description automatically generated

This ensures that session cookies are secure and have an idle timeout to protect against session hijacking.

# 2. Data Encryption and Archival

**Assumptions:**

* The web application is built using ASP.NET or ASP.NET Core.
* User authentication is based on a username and password.
  1. **Insufficient Data Encryption**

Sensitive user data is at risk if stored in plaintext within the database, making it susceptible to unauthorized access or breaches.

**References:**

o OWASP: A6:2017 - Security Misconfiguration

o SANS: CWE-319: Cleartext Transmission of Sensitive Information

**Mitigation Techniques:**

Implement strong encryption algorithms for sensitive data before storing it in the database.

**Implementation:**

1. Use a Strong Encryption Library:

* Choose a reputable encryption library or method suitable for your application.

1. Encrypt Data Before Storage:

* In the code handling data storage, encrypt sensitive fields before persisting them to the database.

A screen shot of a computer

Description automatically generated

Utilize a strong encryption algorithm such as AES to secure sensitive user information.

* 1. **Lack of Secure Archival Process**

Archived data might contain sensitive information, and a lack of a secure archival process can expose this data to unauthorized access.

**References:**

o OWASP: A9:2017 - Using Components with Known Vulnerabilities

o SANS: CWE-272: Least Privilege Violation

**Mitigation Techniques:**

Implement a secure archival process with restricted access to ensure the confidentiality and integrity of archived data.

**Implementation:**

1. Establish Secure Archival Procedures:

* Define strict procedures for archiving data, including encryption, access controls, and periodic reviews.

1. Use Access Controls for Archived Data:

* Apply access controls to limit who can access archived data and what actions they can perform.

A black background with white text

Description automatically generated

This ensures that only authorised personnel can access and manage archived information.

* 1. **Lack of Data Retention Policies**

The absence of clear data retention policies may result in unnecessary data accumulation, increasing the risk of data exposure.

**References:**

o OWASP: A1:2017 - Injection

o SANS: CWE-577: Use of Less Trusted Source

**Mitigation Techniques:**

Implement well-defined data retention policies to govern the lifecycle of stored data, ensuring data is archived or deleted appropriately.

**Implementation:**

1. Define Data Retention Policies:

**-** Establish policies specifying how long different types of data should be retained.



2. Automate Data Archival or Deletion:

* Implement automated processes to archive or delete data according to the defined retention policies.

A black background with white text

Description automatically generated

This reduces the risk of unauthorised access to outdated or unnecessary data that could be sensitive.

# References

1. OWASP Top 10 - <https://owasp.org/www-project-top-ten/>
2. CWE - [CWE - Common Weakness Enumeration - MITRE](https://cwe.mitre.org/)