Student Full Name: Ang Kiat Heng, Alan

Admin No: 222660P

Tutorial Group: IT2163-04

I have chosen the following for my assignment:

* Login Page
* Forget Password or account recovery process.

Table of Contents

[1. Login Page 3](#_Toc152429282)

[**Security Issue #1** 3](#_Toc152429283)

[**Security Issue #2** 4](#_Toc152429284)

[**Security Issue #3** 5](#_Toc152429285)

[2. Forget Password or Account Recovery Page 7](#_Toc152429286)

[**Security Issue #1** 7](#_Toc152429287)

[**Security Issue #2** 9](#_Toc152429288)

[**Security Issue #3** 9](#_Toc152429289)

# 1. Login Page

## **Security Issue #1**

The first security issue would be that of Brute Forcing. Brute Forcing is the attempt to discover a password by systematically trying out every single possible combination of letters, numbers and symbols. This would happen until they have finally found the correct password.

A brute force attack usually starts with a dictionary of words that are familiar to the user or familiar such as family names, pets, favourite locations, etc. (Source: [OWASPS Blocking Brute Force Attack](https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks))

Mitigation Technique #1

Account Lockout

One way would be to block the user from login into the account after multiple unsuccessful attempts at logging in, be it the wrong password or email address. The way to block this could either be locking the account, which is the most common way to do an Account Lockout. However, there is another way, which is to do an IP ban. This would ensure that one person would not be able to access the account or any other account from the same position and device. This would prevent the user from being able to brute force into the account and any other account that they may try to access.

Some of the ways to achieve this would be to have a counter for a particular account to check how many times they have attempted to log into the account. If the counter exceeds the threshold in a certain amount of time, the account will be disabled and the user would have to contact the administrators to unlock the account.

The process to activate the Account Lockout is in fact not that hard. Using ASP.NET identity, you could add an “AccessFailedCount” inside the User and have that check the number of failed attempts. Afterwards, we can add the `LockoutEnabled` Property and the `LockoutEndDateUtc` Property to lock the account out.

An example would be:

However, the issue with an account lockout is that an attacker could cause a Denial Of Service (DoS) by locking out a large number of accounts. This could also lead to a diversion by locking out many accounts and flooding the help desk with support calls.

## **Security Issue #2**

The second security issue would be Weak Password Practices, both on the user and server side. On the server side, what would be considered a weak password practice would be when the passwords of users are stored in plaintext in the database. This is a major security flaw as if the database is compromised, attackers would immediately be able to know what the passwords to the user’s accounts are.

On the user side, if there is a weak password policy, attackers would easily be able to guess the password or use methods such as Brute Force attacks to hack into the user’s account.

Mitigation Technique #1

Stronger Password Policy

As the name implies, implementing a stronger password policy would help in mitigating this security issue. A stronger password policy would include increased password lengths, complexity, amount of times a password can be reused and how old a password can be before needing to be changed. Additional policies which may guard against Brute Force attacks would be to ensure that users are not able to use names and number patterns from their email address or user login ID. This would ensure that the user’s passwords are safe and secure and that even if their password may have been compromised in the past or on another site, their account here might have a different password. (Source: [OWASP Testing for Weak Password Policy](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy))

An example of ensuring a strong password policy is enforced can be done with the help of Regular Expressions (RegEx). Using RegEx, we can ensure users follow the password policies that are implemented.

["^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$"](https://stackoverflow.com/questions/19605150/regex-for-password-must-contain-at-least-eight-characters-at-least-one-number-a) is a Regular Expression statement that ensures users must have a minimum of eight characters, at least one uppercase letter, one lowercase letter, one number and one special character. With this, the user would be less susceptible to hacking.

Mitigation Technique #2

Password Encryption

Using password encryption, this would ensure that even when the database is compromised, the compromised passwords would not be so easily stolen and used. Encryption can be done in a multitude of ways, such as using libraries. These libraries would help to encrypt the passwords and checks the passwords using a function during login, making it so that the decrypted password is not shown in logs.

An example of these encryption libraries are Bcrypt:



The code example above is used in ASP.NET Core to encrypt user passwords when they are registering for an account on the website. This would ensure that the passwords that are stored in the database are encrypted and even if they were compromised, hackers would not be able to easily get the user’s passwords.

## **Security Issue #3**

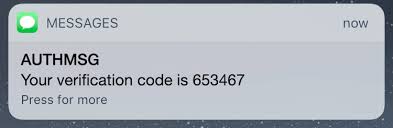
The third security issue would be the lack of or ineffective multi-factor authentication. When there is no multi-factor authentication, hackers might be able to access the user’s account once they have managed to find the password of the user. Furthermore, when there is a lack of multi-factor authentication, users who fall for phishing attacks will immediately give the hacker access to their account. (Source: [Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/))

Mitigation Technique #1

Implementing Two-Factor Authentication (2FA)

Implementing 2FA is a simple and extremely effective technique against phishing attacks and account takeovers. There are many ways in which a website can use 2FA. Some common examples are using codes being sent to emails or phone numbers (Apple ID login, Social Media login) or using an authentication app such as Microsoft Authenticator or Google Authenticator, where a code is provided for verification of the user. Another way of implementing 2FA should the website have a mobile app is to send a notification to the user’s mobile phone via the app to authenticate the user (Github) or tap “Yes” on their mobile devices (Google login 2FA). This will ensure security for the user’s account as the attackers are now not able to access the user’s account unless they have the authentication code.

Example of how a code being sent to a phone number for authentication:



# 2. Forget Password or Account Recovery Page

## **Security Issue #1**

The first security issue is that of Weak Password Reset Mechanisms. Generally, when one wants to reset their password, they can go through a few methods, one of them being Knowledge-Based Answers (KBAs). The issue, however, is that KBAs are often easily guessable or are already obtainable for attackers, making KBAs an unreliable means of verifying user identity.

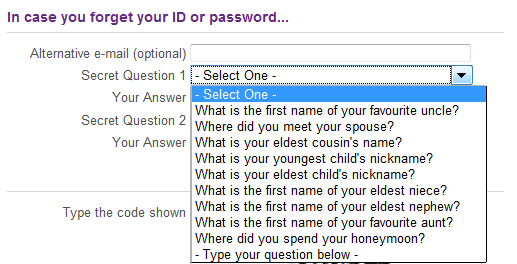
Another method that is widely used is sending password reset links through email. Users are prompted to enter a new password on the site after clicking on the link. However, some issues faced might be that the link sent could be used multiple times, which will allow attackers to have a persistent backdoor to the user’s account. Furthermore, if tokens are unencrypted or easily guessable, attackers might be able to guess or even brute force a token to change the password of the account.

Mitigation Technique #1

Allowing users to write their own security answers

Allowing users to write their own security answers will allow users to be able to choose very strong and unique questions that only users would be able to answers. However, there is also a chance that users would pick a rather weak password, therefore there are some safeguards to be implemented. Firstly, the user should not be able to put their own name or parts of their email address inside their password. Furthermore, the users should not be able to put their current password and common strings such as “password” or “123”. (Source: [OWASP Cheatsheet “Choosing and Using Security Questions”](https://cheatsheetseries.owasp.org/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.html))

An example of KBAs:



Mitigation Technique #2

Encryption of tokens and expiration of tokens.

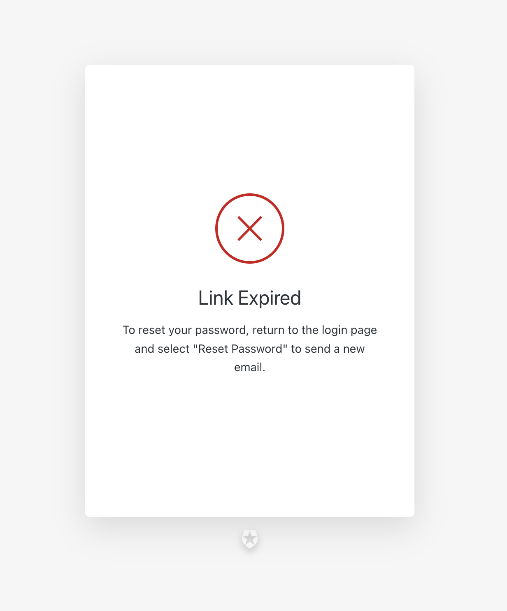
In order to protect against brute forcing and guessing of password reset tokens, tokens should be long and random. This can be done using libraries which provide cryptographically signed tokens such as Json Web Token (JWT) or using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) to generate a token. This would ensure that the tokens are not easy to brute force and would therefore deter attacks. Furthermore, important details such as the user’s ID should not be easily detectable in the URL, as this would allow the attackers to try and change the passwords of other accounts using the same token. Therefore, the user’s ID should either be well hidden or in the case of a JWT, the user’s ID can be hidden inside the JWT. (Source: [OWASP “Testing for Weak Password Change or Reset Functionalities”](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities))

An example of this can be seen below:



Secondly, password reset links should have an expiry date and can only be used once. This is to ensure that attackers do not have a consistent backdoor to the user’s account. The tokens provided to the user should usually be, at most, valid for 1 hour and no more and are only for one-time use. (Source: [OWASP “Testing for Weak Password Change or Reset Functionalities”](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities))

What a user should see when a token/link is expired:



## **Security Issue #2**

The next security issue is concerning Insecure Communication Channels. Even if there are encrypted token and link expiry timings, it would not matter if the password reset link is compromised whilst being sent to the user. Once the password reset link is compromised, the attackers would be able to reset the user’s password, thus granting him access to the user’s account.

Mitigation Technique #1

Applying of SSL/TLS to transport channels

Usage of the HTTPS protocol, which utilizes SSL/TLS protocols, is a safe and secure way to transport sensitive data over to the user and vice versa. This would ensure that attackers are not able to intercept or eavesdrop on the password reset link being sent to the user. Furthermore, the private key used by the server should not be easily guessable. If possible, it is best practice to not use keywords such as the website name but instead use a random generator to generate a private key of at least 2048 bits. The private key should also be protected from unauthorized access using filesystem permissions and other technical and administrative controls.

Additionally, certificates should use hashing algorithms such as SHA-256 rather than older algorithms such as MD5 and SHA-1 as these are outdated and have already been reverse engineered. (Source: [OWASP Transport Layer Protection Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html))

## **Security Issue #3**

The third security issue would be the lack of Multi-Factor Authentication (MFA). Should a case where the password reset link is compromised, the lack of MFA would allow the attackers to be able to easily change the user’s password without any resistance.

Mitigation Technique #1

Implementing Multi-Factor Authentication (MFA)

Using MFA, when the attacker tries to change the password of the account, they would be prompted to enter a code which is sent via phone number or found inside an authentication application. This would prevent the attackers from being able to successfully change the user’s password. MFA is extremely easy to implement and can be done in many different ways. Some common examples are using codes being sent to phone numbers (Apple ID login, Social Media login) or using an authentication apps (Google Authenticator, Microsoft Authenticator), where a code is provided for verification of the user. Another way of implementing MFA should the website have a mobile app is to send a notification to the user’s mobile phone via the app to authenticate the user (Github) or tap “Yes” on their mobile devices (Google login 2FA). This will ensure security for the user’s account as the attackers are now not able to access the user’s account unless they have the authentication code.

Furthermore, the website could also provide the users emergency codes upon account creation which could be used to verify themselves when they are trying to reset their password. These emergency codes are often times a randomised string which includes letters and numbers. This method would mean that even if all other options of MFA are compromised, the user would still be able to prove themselves via these emergency codes. The only issue with using emergency codes is that should these codes be easy to guess or are short, they can easily be bypassed using Brute Force attacks. Therefore, the best option would be to have a timeout period once they have entered the emergency code incorrectly after a certain number of times. In addition, these emergency codes should not be used more than once and once it is used, the emergency code would then be invalidated.

# References

OWASP Blocking Brute Force Attacks:

<https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks>

OWASP Testing for Weak Password Policy:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy>

Password Regex Code:

<https://stackoverflow.com/questions/19605150/regex-for-password-must-contain-at-least-eight-characters-at-least-one-number-a>

OWASP A07:2021 Identification and Authentication Failures:

<https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/>

OWASP Cheatsheet “Choosing and Using Security Questions”:

<https://cheatsheetseries.owasp.org/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.html>

OWASP “Testing for Weak Password Change or Rest Functionalities”:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities>

OWASP “Transport Layer Protection Cheat Sheet”:

<https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html>