**Name: Emon Monsur Student Number: 121367643**

**Question 1 – Password Guessing:**

As it is stated that password can be 6-8 characters, we need to, for each of these lengths, calculate the possible number of passwords.

For six characters:

Passwords can either contain alphabetical characters (26 possible characters) or integers (10 possible characters). For a total of thirty-six possible characters.

So possible combinations.

However it is also stated that passwords must contain at-least one alphabetic and one numeric character. So we must disregard passwords that contain only alphabetic or only numeric characters.

Thus possible passwords for passwords of length 6.

For all passwords lengths:

For passwords of length 7 and 8 characters, all we need to do is replace six with 7 and 8 to get passwords of each of those respective lengths, so total number of possible passwords is:

) + ) + ), which is equal to 2,684,372,063,360.

**Question 2 – Web Authentication:**

JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types.

JSON Web Tokens consist of three parts separated by dots:

* Header: Contains the metadata like type and algorithm used.
* Payload: Contains the claims, which are statements about the user and additional data. Claims come in three types: registered, public, and private.
* Signature: A hashed value generated by signing the header and payload using a secret key, which is used to verify the token’s integrity and authenticity.

It is base64 encoded.

The various parts of the given JWT are:

* Header: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
* Payload: eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ
* Signature: SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

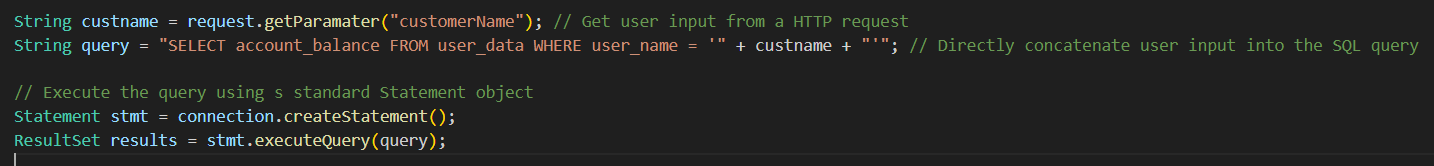
We are able to recognise the different parts by separating them according to the dots found within.

**Question 3 – Prepared Statement:**

SQL injection attacks exploit vulnerabilities where user input is directly included in a SQL query. Prepared statements are used to separate SQL code from user input, which ensures that the databases treat input as data and not code.

Without Prepared Statements:

When an application directly embeds user input into an SQL query string, which callows attackers to inject malicious SQL commands.



If the user input is malicious, for example ‘ OR ‘1’=1, this query will always be true and may expose sensitive data.

With Prepared Statements:

User input is treated as data here, and not executable code – this is because we use placeholders (?) instead of directly concatenating input.

A black screen with text

Description automatically generated

The database engine ensures that user input has properly escaped and not executed as SQL. No direct query modification – this is because the database receives the query and data separately, so even if the input includes SQL syntax, it cannot alter the query.

**Question 4 - ECB :**

Electronic Code Book (ECB) is a simple mode operation with a block cypher that is mostly used with symmetric key encryption. The input plaintext is broken into numerous blocks – these blocks are individually encrypted using the encryption key. As a result, each encrypted block can also be decrypted individually.

Some reasons why ECB encryption mode should not be used to encrypt large amounts of data include:

* ECB uses simple substitution so two identical blocks of plaintext result in two identical blocks of ciphertext, and since identical blocks produce identical ciphertext, attackers can detect repeated data patterns and even replace or replay ciphertext blocks to manipulate encrypted data.
* There is no randomisation in ECB – it does use an initialisation vector, meaning the encryption of the same plaintext results in the same ciphertext. This makes it easy for attackers to recognise patterns.

**Question 5 - XSS:**

With stored XSS attacks, the application stores un-sanitised user input that is viewed later by another user or an administrator. Or in more general terms, an attacker may log into a forum and post an entry that contains a script that is not visible in the interface (as its executed as JavaScript code). A user logs into the forum, and the forum entry is displayed alongside the script, and using this script, the user’s access token will be extracted.

And with reflected XSS attacks, the application includes unvalidated and unescaped user input as part of HTML output. A successful attack can allow the attacker to execute arbitrary HTML and JavaScript in the victim’s browser. For reflected XSS attacks, the user will need to interact with a malicious link pointing to an attacker-controlled page.

So the difference between the two is with reflected XSS, the malicious script is not stored on the server, and instead, it is embedded in a URL that I send to the victim. Whereas with stored XSS, the script is stored permanently on the target server. Stored XSS attacks can attack multiple users, whereas reflected XSS is targets a single user. Stored XSS is used to steal session cookies, credentials, or the like, and reflected XSS is used in phishing attacks or to trick users into executing malicious scripts.