Assesment-4(Thursday)

Solution for problem statement of CS3:

The main objectives of the problem of the problem statement is that:

- Objective 1 (Simulate the breach) → The program should generate a list of usernames and passwords.
- Objective 2 (Identify vulnerabilities) → The program should analyse how passwords are stored and detect weak storage methods.
- Objective 3 (Evaluate password strength) → The program should assess how strong or weak each password is and categorize them.

Now, input:

- A text file (credentials.txt) containing a list of usernames and passwords.

  Each line of the file follows the format:

  Username: password…like,

user1: password123

user2: Password123!

user3: weak-password

- Example Input (credentials.txt):

Ananya_anu: Anu@2006

Geetahnjali_H: Geeanju666

Jyothi_p: jyo!56

Output:

1. A list of compromised usernames and passwords.

   Ananya_anu: Anu@2006

   Geetahnjali_H: Geeanju666

   Jyothi_p: jyo!56 ……etc-> display the lists.

2. A report identifying potential vulnerabilities in Cloudstrike's password storage system

   This section analyzes how the passwords were stored and identifies weaknesses in Cloud Strike's security practices.

Assesment-4(Thursday)

If an attacker can retrieve and view user passwords as plaintext (like in credentials.txt), it means Cloud Strike is storing passwords without encryption or hashing, making it easy for attackers to misuse them.

The vulnerabilities might include:

1. Storage of Plaintext Passwords

2. Weak Password Policy

3.Lack of Multi-Factor Authentication (MFA)

4.Predictable Password Patterns

5.No Salting or Hashing (Assumed)……etc.

Requirements:

1. Use C programming language.

2. Use file input/output operations to read and write data. (Example Input (credentials.txt):)

3. Use string manipulation functions to process usernames and passwords.

[Usage string manipulation functions like strlen(), strstr()…etc.]

4. Implement a simple password strength evaluation algorithm (e.g., check for length, uppercase, lowercase, digits, and special characters).

- The program must classify passwords as Weak, Moderate, or Strong based on these rules:

✓ Weak: Less than 8 characters, or lacks variety (e.g., only lowercase letters).
✓ Moderate: At least 8 characters, contains two types (uppercase, lowercase, digits, or special characters).
✓ Strong: At least 12 characters and contains a mix of uppercase, lowercase, digits, and special characters.

CODE:

```
#include <stdio.h>

#include <string.h>

#include <ctype.h>
```

Assesment-4(Thursday)

```
#define MAX_USERS 3



char credentials[MAX_USERS][2][50] = {
   {"Ananya_anu", "Anu@2006"},
   {"Geetahnjali_H", "Geeanju666"},
   {"Jyothi_p", "jyo156"}
};



const char* evaluate_password(const char* password) {
   int length = strlen(password);
   int has_upper = 0, has_lower = 0, has_digit = 0, has_special = 0;

   for (int i = 0; i < length; i++) {
      if (isupper(password[i])) has_upper = 1;
      else if (islower(password[i])) has_lower = 1;
      else if (isdigit(password[i])) has_digit = 1;
      else has_special = 1;
   }

   if (length >= 12 && has_upper && has_lower && has_digit && has_special)
      return "Strong";
   if (length >= 8 && ((has_upper + has_lower + has_digit + has_special) >= 2))
      return "Moderate";
   return "Weak";
}
```

Assesment-4(Thursday)

```
void check_vulnerabilities(const char* username, const char* password) {
    if (strlen(password) < 8) {
        printf("- Weak password: Less than 8 characters\n");
    }
    if (strstr(password, "123") || strstr(password, "password")) {
        printf("- Predictable password pattern detected\n");
    }
    if (strstr(password, username)) {
        printf("- Password contains username (high risk)\n");
    }
}

int main() {
    printf("Compromised Credentials:\n");
    for (int i = 0; i < MAX_USERS; i++) {
        printf("%s: %s\n", credentials[i][0], credentials[i][1]);
    }

    printf("\nPassword Strength Evaluation:\n");
    for (int i = 0; i < MAX_USERS; i++) {
        printf("%s: %s (%s)\n", credentials[i][0], credentials[i][1],
evaluate_password(credentials[i][1]));
    }

    printf("\nIdentified Vulnerabilities:\n");
    for (int i = 0; i < MAX_USERS; i++) {
        printf("%s: \n", credentials[i][0]);
        check_vulnerabilities(credentials[i][0], credentials[i][1]);
    }
```

```
    return 0;
}
```

OUTPUT:

Compromised Credentials:

Ananya_anu: Anu@2006

Geetahnjali_H: Geeanju666

Jyothi_p: jyo156

Password Strength Evaluation:

Ananya_anu: Anu@2006 (Moderate)

Geetahnjali_H: Geeanju666 (Moderate)

Jyothi_p: jyo156 (Weak)

Identified Vulnerabilities:

Ananya_anu:

Geetahnjali_H:

Jyothi_p:

- Weak password: Less than 8 characters