Course Name:	**Information and Cyber Security Laboratory**	**Semester:**	**VII**
Date of Performance:		**Batch No:**	**B - 1**
Faculty Name:	**Prof. Makarand Kulkarni**	**Roll No:**	**16014022050**
Faculty Sign & Date:		**Grade/Marks:**	**25**

Experiment No: 8

Title: Implementation of Password Policy

Aim and Objective of the Experiment:

Implementation of Password Policy

COs to be achieved:

CO 4. Understand effective countermeasures to mitigate risk of cyber threats and crime.

Books/Journals/Websites referred:

- **OWASP Password Policy Cheat Sheet**
- **NIST Digital Identity Guidelines**
- **Python Official Documentation**

Tools required:

Python code

Theory:

A password policy is a set of rules to enhance account security by enforcing strong passwords, protects systems from unauthorized access, brute-force attacks, and other cybersecurity threats. The ord policy helps ensure authentication security, reduce password compromises, and maintain m standards that comply with regulatory requirements.

A password policy is a set of rules and best practices designed to enhance the security of user accounts by enforcing the creation and management of strong passwords. Implementing a robust password policy helps protect sensitive data and systems from unauthorized access, brute-force attacks, and other cybersecurity threats.

Objectives:

- To ensure secure user authentication.
- To minimize the risk of compromised passwords.
- To enforce uniform password standards across all users.
- To comply with organizational or regulatory security requirements.

Implementation details:

Key Components of a Password Policy

A well-implemented password policy typically includes the following elements:

1. **Password Length:**
   Minimum of 8–12 characters to ensure complexity.
2. **Password Complexity:**
   Must include a combination of:
   - Uppercase letters (A–Z)
   - Lowercase letters (a–z)
   - Numbers (0–9)
   - Special characters (e.g., @, #, $, %)
3. No space or hyphen is allowed in the password
4. **Password Type: Weak or Strong password**

Output / Screenshots:

```
import re

def check_password_policy(password):
    """Check whether a password complies with the defined policy."""

    # Check length: minimum 8 characters
    if len(password) < 8:
        return "Password too short. Minimum length is 8."
```

```
    # Check for spaces or hyphens
    if ' ' in password or '-' in password:
        return "Password must not contain spaces or hyphens."

    # Check for uppercase, lowercase, digit, special character
    if not re.search(r'[A-Z]', password):
        return "Password must include at least one uppercase letter."
    if not re.search(r'[a-z]', password):
        return "Password must include at least one lowercase letter."
    if not re.search(r'[0-9]', password):
        return "Password must include at least one digit."
    if not re.search(r'[!@#$%^&*(),.?":{}|<>]', password):
        return "Password must include at least one special character."

    # Classify password strength
    length_score = len(password) >= 12
    categories = sum([bool(re.search(pat, password)) for pat in [r'[A-Z]', r'[a-
z]', r'[0-9]', r'[!@#$%^&*(),.?":{}|<>]']])

    if length_score and categories == 4:
        return "Strong password."
    else:
        return "Weak password (meets minimum requirements but can be stronger)."

# Test examples
passwords = ["Pass123!", "Password123!", "P@ssword1", "P@ss word1", "P@ss-Word1"]
for pwd in passwords:
    print(f"Password: '{pwd}' --> {check_password_policy(pwd)}")
```

```
Password: 'Pass123!' --> Weak password (meets minimum requirements but can be stronger).
Password: 'Password123!' --> Strong password.
Password: 'P@ssword1' --> Weak password (meets minimum requirements but can be stronger).
Password: 'P@ss word1' --> Password must not contain spaces or hyphens.
Password: 'P@ss-Word1' --> Password must not contain spaces or hyphens.
```

Post Lab Subjective/Objective type Questions:

1. **Why is it important to enforce password expiration and history in a system?**

Enforcing password expiration ensures users change passwords periodically, reducing the risk of compromised credentials being misused long-term. Password history prevents reuse of old passwords, making it harder for attackers to exploit previously known passwords and increasing overall security.

2. **Describe the steps involved in implementing a password policy in an organization. Include examples from system configurations such as Windows or Linux.** Steps to implement a password policy include:
3. Defining password requirements (length, complexity, expiration, history).
4. Configuring system settings to enforce rules.
5. In **Windows**, use Group Policy Editor to set password policies like minimum length, complexity, maximum password age, and password history enforcement.
6. In **Linux**, edit /etc/login.defs or use PAM modules (/etc/pam.d/common-password) to enforce password complexity and expiration.
7. Educating users about password requirements and best practices.
8. Monitoring compliance and updating policies as threats evolve.

Conclusion:

The experiment successfully demonstrated the importance of implementing a strong password policy to enhance cybersecurity. Through Python programming, the password policy was enforced with rules on length, complexity, and prohibited characters, classifying passwords as weak or strong. Such policies help protect systems from unauthorized access and cyber threats by encouraging secure authentication practices

Signature of faculty in-charge with Date: