MICRO PROJECT

PROJECT

DONE BY,

KIRUTHIVINYA I

Brute-Force Attack Simulation on a Login System

1. Objective

To simulate how a brute-force attack works on a simple login system and understand how delays, lockouts, and security measures help prevent real-world attacks.

2. Technologies Used

- Python
- Time module
- Random module (for demo)

3. Program 1: Simple Login System (Vulnerable)

Python

```
#Simple vulnerable login system (NO PROTECTION)
correct_username = "admin"
correct_password = "1234"
def login(user. pwd):
if user == correct_username and pwd ==
correct_password:
return True
return False
```

4. Program 2: Brute Force Attack Simulation

Python

```
#Simulate brute force attack
import time
possible_passwords = ["1111". "2222". "1234". "0000". "5678"]
print("Starting Brute Force Attack...\n")
for attempt, pwd in enumerate(possible_passwords. start=1):
print("Trying password (pwd) (Attempt (attempt))")
time.sleep(0.5) # small delay for demonstration
if pwd == "1234":
print("\n Password FOUND", pwd)
break
else:
print("Password not ")
```

5. Expected Output (Vulnerable System)

```
Starting Brute Force Attack...
Trying password 1111 (Attempt 1)
Trying password 2222 (Attempt 2)
Trying password 1234 (Attempt 3)
Password FOUND: 1234
This shows how simple passwords are quickly cracked.
```

6.Program 3: Login System with Delay + Account Lockout (Protection)

```
# Secure login system with protection

correct_username = "admin"
correct_password = "1234"
attempts = 0
max_attempts = 3
lockout_time = 5  # seconds

def secure_login(user, pwd):
    global attempts

    if attempts >= max_attempts:
        print(f"Account locked! Try again in {lockout_time} seconds.")
        time.sleep(lockout_time)
        attempts = 0  # reset after lockout

    if user == correct_username and pwd == correct_password:
        print("Login Successful!")
        attempts = 0
        return True
    else:
        attempts += 1
        print("Login Failed!")
        print("Attempts left:", max_attempts - attempts)
        return False
```

07. Brute-Force Attack Fails Due to Lockout

Example Output:

Login Failed!

Attempts left: 2

Login Failed!

Attempts left: 1

Login Failed!

Attempts left: 0

Account locked! Try again in 5 seconds.

Now the bot cannot try unlimited passwords Brute-force blocked.

7. Sample Output

Without Security (Fast Password Crack)

Trying password 1111

Trying password 2222

Trying password 1234

Password FOUND: 1234

With Security (Attack Stopped)

Login Failed! Attempts left: 2

Login Failed! Attempts left: 1

Login Failed! Attempts left: 0

Account locked! Try again in 5 seconds.

8. Log File Example

Password tried: 1111 → Failed

Password tried: 2222 → Failed

Password tried: 1234 → Success

9. Results and Findings

- Weak passwords are easily cracked
- Brute-force attacks depend on unlimited attempts
- Adding delay + lockout makes attacks slow and difficult
- Strong passwords and monitoring greatly improve security

---

10. Impact of Brute-Force Attacks

- Passwords like 1234, 1111, password are cracked in seconds
- Attackers use list of thousands of passwords
- Without delay/lockout, password can be broken easily
- Automated bots can try thousands of combinations per minute

11. Countermeasures

- Strong passwords (min 8-12 characters)
- Add login delay (0.5-2 seconds)
- Account lockout after 3-5 failures
- Captcha
- Two-Factor Authentication

- Avoid common passwords
- Monitor login logs for suspicious activity

12. Conclusion

This micro project successfully demonstrates how brute-force attacks work and how simple security mechanisms can protect a login system.
A combination of strong passwords, delays, lockouts, and logging provides effective defense against such attacks.

13. Future Enhancements

- Add CAPTCHA
- Use hashed passwords instead of plain text
- Integrate Two-Factor Authentication (2FA)
- Allow login from database instead of fixed values