2nd task_Password Strength Analysis Tool Documentation

Overview

This project consists of a set of tools designed to analyze password strength by calculating the time required to brute-force passwords of varying lengths. The system includes:

1. A C++ program to calculate password combinations and crack time
2. A Bash script to create test users with progressively longer passwords
3. A Python script to clean up test users

Components

1. calculate.cpp - Password Strength Calculator

Purpose: Calculates the number of possible password combinations and estimates brute-force cracking time.

Features:

- Calculates total combinations based on character set size (96 characters: A-Z, a-z, 0-9, and special characters)
- Estimates cracking time given a hash rate
- Provides output in human-readable format

Usage:

```
./calculate <password_length> <hashes_per_second>
```

Parameters:

- `password_length`: Length of the password to analyze (must be > 0)
- `hashes_per_second`: Estimated hash operations per second that an attacker can perform

Output:

- Total possible combinations for the given password length
- Estimated time to brute-force the password (in hours or "Less than hour")

2. hash.sh - Test User Creation Script

Purpose: Automates creation of test users with progressively longer passwords and analyzes their strength.

Features:

- Creates 10 test users (user1 to user10)
- Generates random passwords for each user (length 1-10 characters)
- Automatically analyzes each password's strength using calculate.cpp
- Requires root privileges to create users and set passwords

Usage:

```
sudo ./hash.sh <hashes_per_second>
```

Parameters:

- `hashes_per_second`: Hash rate to use for password strength analysis

Output:

- Creates users with displayed passwords
- Shows password strength analysis for each user

3. python_user_removel.py - Cleanup Script

Purpose: Removes all test users created by hash.sh.

Features:

- Removes users user1 through user10
- Deletes home directories (-r flag)
- Simple Python implementation using system commands

Usage:

```
python3 python_user_removel.py
```

Technical Details

Password Combination Calculation

The system uses the formula:

```
Combinations = charset_size^password_length
```

Where charset_size is 96 (26 uppercase + 26 lowercase + 10 digits + 34 special characters).

Time Estimation

The crack time is calculated as:

```
Total_seconds = Combinations / hashes_per_second
```

The output is then converted to hours for readability.

Security Considerations

1. The test script displays generated passwords in plaintext - this should only be used in testing environments
2. The character set size (96) can be modified in calculate.cpp if needed
3. The system assumes brute-force attacks can try all combinations sequentially

Example Usage

1. Analyze a password of length 8 with 1 billion hashes/second:

```
./calculate 8 1000000000
```

1. Create test users and analyze their passwords with 1 million hashes/second:

```
sudo ./hash.sh 1000000
```

1. Clean up test users:

```
python3 python_user_removel.py
```

Limitations

1. Doesn't account for dictionary attacks or common password patterns
2. Assumes constant hash rate without hardware limitations
3. Time estimation doesn't include factors like network latency or system throttling

Future Enhancements

1. Add support for different character sets

2. Implement more sophisticated time estimation (days, years)
3. Add graphical output or visualization
4. Incorporate common password patterns into strength calculation

This documentation provides a comprehensive overview of the password strength analysis toolset. The system is particularly useful for educational purposes to demonstrate the importance of password length and complexity in security.