Name - HARSHIL AMIT BUCH

Class - TY CSF-1

Roll Number - 14

Experiment No. 2

Title:

Use a Password Strength Tester to Evaluate the Strength of Various Passwords

Objective:

To understand the importance of strong passwords and evaluate different types of passwords using tools like `cracklib-check`, `pwscore`, and `zxcvbn` on Kali Linux.

Software/Tools Required:

- Kali Linux
- Terminal
- Tools: `cracklib-runtime`, `libpam-pwquality`, `zxcvbn` (Node.js)

Theory:

What is Password Strength?

Password strength is a measure of how resistant a password is to guessing and brute-force attacks. Strong passwords are:

- At least 12–16 characters long
- Contain a mix of uppercase, lowercase, numbers, and special characters
- Avoid common words, names, or predictable patterns
- Not reused from other accounts

Common Password Evaluation Tools in Kali Linux:

Installation Commands:

```
bash
CopyEdit
sudo apt update
sudo apt install libcrack2 cracklib-runtime libpam-pwquality nodejs npm
npm install -g zxcvbn
```

Procedure:

1. Using `cracklib-check`:

```
bash
CopyEdit
echo "Password123" | cracklib-check
```

Output:

```
Password123: it is based on a dictionary word
```

2. Using `pwscore`:

```
bash
CopyEdit
echo "D3f!AnT#92" | pwscore
```

Output:
A numeric score out of 100.

3. Using `zxcvbn`:

```
bash
CopyEdit
zxcvbn "Welcome@2025"
```

Output:
Shows password score (0–4), estimated crack time, and feedback.

Sample Passwords and Evaluation:

Password	Tool Used	Strength Score	Comments
`123456`	cracklib/ pwscore	Very Weak (5)	Common, predictable
`password`	cracklib/zxcvbn	Very Weak (0)	Dictionary word
`Password123`	zxcvbn	Weak (1)	Simple pattern
`Pa$ $w0rd2025!`	pwscore	Moderate (45)	Decent mix, still somewhat predictable
`Zx!7@qP#v9D3`	zxcvbn	Very Strong (4)	Long, complex, high entropy
`iloveyou`	cracklib	Very Weak	Common phrase

Output Screenshots:

```
┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# echo "123456" | cracklib-check
123456: it is too simplistic/systematic

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# echo "123456" | pwscore
Password quality check failed:
 The password is shorter than 8 characters

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─#
```

```
┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# echo "password" | cracklib-check
password: it is based on a dictionary word

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# zxcvbn --user-input
bash: zxcvbn: command not found

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# zxcvbn --user-input
usage: zxcvbn [-h] [--user-input USER_INPUT] [--max-length MAX_LENGTH]
zxcvbn: error: argument --user-input: expected one argument

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# zxcvbn --user-input ""
Password:
{
  "password": "password",
  "guesses": "3",
  "guesses_log10": 0.4771212547196624,
  "sequence": [
    {
      "pattern": "dictionary",
      "i": 0,
      "j": 7,
      "token": "password",
      "matched_word": "password",
      "rank": 2,
      "dictionary_name": "passwords",
      "reversed": false,
      "l33t": false,
      "base_guesses": 2,
      "uppercase_variations": 1,
      "l33t_variations": 1,
      "guesses": 2,
      "guesses_log10": 0.30102999566398114
    }
  ],
  "calc_time": "0:00:00.001762",
  "crack_times_seconds": {
    "online_throttling_100_per_hour": "108.00000000000005995204333O",
    "online_no_throttling_10_per_second": "0.3",
    "offline_slow_hashing_1e4_per_second": "0.0003",
    "offline_fast_hashing_1e10_per_second": "3E-10"
  },
  "crack_times_display": {
    "online_throttling_100_per_hour": "2 minutes",
    "online_no_throttling_10_per_second": "less than a second",
    "offline_slow_hashing_1e4_per_second": "less than a second",
    "offline_fast_hashing_1e10_per_second": "less than a second"
  },
  "score": 0,
  "feedback": {
    "warning": "This is a top-10 common password.",
    "suggestions": [
      "Add another word or two. Uncommon words are better."
    ]
  }
}

┌──(myenv)(root㉿ 3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─#
```

```
┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# zxcvbn --user-input ""
Password:
{
  "password": "Password123",
  "guesses": "1191",
  "guesses_log10": 3.075911761482777,
  "sequence": [
    {
      "pattern": "dictionary",
      "i": 0,
      "j": 10,
      "token": "Password123",
      "matched_word": "password123",
      "rank": 595,
      "dictionary_name": "passwords",
      "reversed": false,
      "l33t": false,
      "base_guesses": 595,
      "uppercase_variations": 2,
      "l33t_variations": 1,
      "guesses": 1190,
      "guesses_log10": 3.07554696139253
    }
  ],
  "calc_time": "0:00:00.005504",
  "crack_times_seconds": {
    "online_throttling_100_per_hour": "42876.000000000002380096120l9",
    "online_no_throttling_10_per_second": "119.1",
    "offline_slow_hashing_1e4_per_second": "0.1191",
    "offline_fast_hashing_1e10_per_second": "1.191E-7"
  },
  "crack_times_display": {
    "online_throttling_100_per_hour": "12 hours",
    "online_no_throttling_10_per_second": "2 minutes",
    "offline_slow_hashing_1e4_per_second": "less than a second",
    "offline_fast_hashing_1e10_per_second": "less than a second"
  },
  "score": 1,
  "feedback": {
    "warning": "This is a very common password.",
    "suggestions": [
      "Add another word or two. Uncommon words are better.",
      "Capitalization doesn't help very much."
    ]
  }
}
```

```
┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# echo "Pa$$w0rd2025!" | pwscore
90

┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─#
```

```
┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# zxcvbn --user-input ""
Password:
{
  "password": "Zx!7@qP#v9D3",
  "guesses": "1000000000001",
  "guesses_log10": 12.000000000000433,
  "sequence": [
    {
      "pattern": "bruteforce",
      "token": "Zx!7@qP#v9D3",
      "i": 0,
      "j": 11,
      "guesses": 1000000000000,
      "guesses_log10": 11.999999999999998
    }
  ],
  "calc_time": "0:00:00.003422",
  "crack_times_seconds": {
    "online_throttling_100_per_hour": "36000000000036.00199840144433",
    "online_no_throttling_10_per_second": "100000000000.1",
    "offline_slow_hashing_1e4_per_second": "100000000.0001",
    "offline_fast_hashing_1e10_per_second": "100.0000000001"
  },
  "crack_times_display": {
    "online_throttling_100_per_hour": "centuries",
    "online_no_throttling_10_per_second": "centuries",
    "offline_slow_hashing_1e4_per_second": "3 years",
    "offline_fast_hashing_1e10_per_second": "2 minutes"
  },
  "score": 4,
  "feedback": {
    "warning": "",
    "suggestions": []
  }
}

┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─#
```

```
┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─# echo "iloveyou" | cracklib-check
iloveyou: it is based on a dictionary word

┌──(myenv)(root㉿3ddfedb4-cbf4-40cd-bc80-4b6c7bdaba60)-[/mnt]
└─#
```

Outcome:

By the end of this experiment, students will be able to:

- Identify characteristics of weak vs strong passwords
- Evaluate password strength using multiple tools
- Understand time-to-crack estimates and dictionary-based weakness
- Apply secure password practices in real-world scenarios

Conclusion:

This experiment helped students understand the importance of strong passwords and demonstrated how different tools can analyze and rate password security, helping in developing better security hygiene.