A Customizable Password Generator

MINOR PROJECT REPORT

Submitted by

Aadil(24MCC20077)

in partial fulfillment for the award of the degree of

Master of Computer Applications

Cloud Computing & DevOps

In

University Institute of Computing

Chandigarh University

November 2024

BONAFIDE CERTIFICATE

I certify that this project report, “**A Customizable Password Generator**,”
is the bona fidework of AADIL , who did the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
Dr. Abdullah	**Mr. Rishab Tomar**
HOD	
UIC	**UIC**

Submitted for the project viva voce examination held on **Nov 2024**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Declaration

I at this moment declare that the project report entitled "**A Customizable password Generator"**Submitted by me to the **University Institute of Computing, Chandigarh University, Gharuan,** in partial fulfillment of the requirement for the award of the degree "**Master of Computer Application- Cloud Computing & DevOps**" is a Bonafede project workcarried out by me under the guidance of "**Mr. Rishab Tomar**."
I further declare that the work reported in this project has not been submitted and will not be submitted, either inpart or in full, for the award of any other degree or diploma in this institute or any otherinstitute or university.

Project Guide: Mr. Rishab Tomar **HOD:** Dr. Abdullah

Date: November, 2024

Submitted by:

Aadil (24MCC20077)

UNIVERSITY INSTITUTE of COMPUTING

Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Certificate Of Originality

This is to certify that the project report entitled "A Customaziable password Genrator" submitted by me in partial fulfillment of the requirements for the award of the Degree Master of Computer Application- Cloud Computing & DevOps (MCA CC & DevOps) is a bonafide record of the work carried out under my guidance and supervision at the University Institute of Computing of the Chandigarh University.

Submitted by:

Aadil (24MCC20077)

Acknowledgment

I take immense pleasure in thanking our HOD Dr. Abdullah for permitting me to carry out this project work. I wish to express my deep sense of gratitude to my Guide Mr. Rishab Tomar for his able guidance and useful suggestions, which helped me in completing the project work, in time. Words are inadequate in offering my thanks for his encouragement and cooperation in carrying out the project work. Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents and their blessings, and my friends & classmates for their help and wishesfor the successful completion of this project.

Date: Nov, 2024

Place: University Institute of Computing, Chandigarh University

Submitted by:

Aadil (24MCC20077)

INDEX

Abstract

In an era where digital security is of paramount importance, the role of strong, unique passwords cannot be overstated. With increasing instances of data breaches and unauthorized access, individuals and organizations alike face the pressing need for effective password management. This project, titled **Linux Password Generator**, addresses this need by providing a command-line tool that allows users to generate secure, random passwords tailored to their specific requirements. Developed using Bash scripting, the password generator enables users to define critical parameters such as password length and character complexity.

The script supports customizable options for including uppercase letters, numbers, and special characters, making it versatile enough to accommodate various security policies and personal preferences. By leveraging the built-in capabilities of Linux, the generator eliminates the need for third-party applications, ensuring ease of access and reliability. The implementation includes features for command-line argument parsing, allowing users to specify their desired password attributes in a straightforward manner.

Throughout the development process, the script was rigorously tested to ensure that it generates passwords that meet the expected criteria for strength and randomness. Results indicated that the tool effectively produces complex passwords that significantly enhance security.

This project not only highlights the importance of strong password practices but also showcases the practicality of using Bash scripting for creating useful utilities in a Linux environment. Future enhancements could include integrating cryptographic random number generation, developing a graphical user interface, and enabling password storage functionalities. By empowering users to create secure passwords efficiently, the **Linux Password Generator** contributes positively to personal and organizational security strategies in an increasingly digital world.

1. Introduction

In today's digital landscape, where online threats and cyberattacks are rampant, the importance of secure password management has never been more critical. Weak or reused passwords are often the primary vulnerability exploited by attackers, leading to unauthorized access to sensitive information and systems.

As a response to this growing concern, the need for effective password generation tools has become increasingly apparent.

The **Linux Password Generator** project aims to provide users with a simple yet powerful solution for creating strong, random passwords that adhere to best security practices. Developed as a command-line tool using Bash scripting, the generator empowers users to create passwords tailored to their specific needs, allowing for customization in terms of length and complexity. Users can define parameters such as the inclusion of uppercase letters, numbers, and special characters, which are essential for creating robust passwords that are difficult to guess or crack.

This project leverages the inherent capabilities of Linux to offer a lightweight and efficient utility without the necessity for third-party applications. The focus on command-line usability ensures that it can be easily integrated into various workflows, catering to both technical users and those seeking straightforward solutions to enhance their security posture.

Ultimately, the **Linux Password Generator** not only simplifies the password creation process but also contributes to the broader goal of promoting safer online practices, thereby playing a vital role in safeguarding personal and organizational data in an increasingly interconnected world.

2. Objectives

The primary objective of the **Linux Password Generator** project is to provide a practical, efficient tool for generating strong, customizable passwords within a Linux environment. This project is designed to address key aspects of password security by allowing users to create complex, random passwords based on their specific needs and security requirements. The objectives of this project include:

1. **Enhancing Security**: Generate random passwords that reduce vulnerability to brute-force attacks by increasing password complexity through various character options, such as uppercase letters, numbers, and special characters.
2. **Customization and Flexibility**: Enable users to define password length and select desired character types, allowing them to create passwords that align with specific security policies or personal preferences.
3. **Ease of Use**: Develop a simple command-line interface with intuitive options for length and character customization, making the tool accessible for users of different technical backgrounds.

4. **Portability and Efficiency**: Implement the tool in Bash, leveraging the native capabilities of Linux to create a lightweight, dependency-free solution that operates seamlessly on any Linux distribution.

Overall, this project aims to foster secure password practices by providing an easy-to-use, Linux-native tool that encourages users to adopt stronger, more complex passwords in their everyday digital interactions.

3. Features and Functionality

The **Linux Password Generator** offers a range of customizable features to help users create strong, secure passwords suited to their needs. Key features include:

1. **Password Length Customization**: Users can specify the desired length of the password, accommodating various security requirements from shorter passwords for casual use to longer, high-security passwords.
2. **Character Type Options**: The tool allows users to include or exclude specific character types:
   - **Lowercase Letters**: Base character set, included by default.
   - **Uppercase Letters**: Optional inclusion for added complexity.
   - **Numbers**: Optional inclusion to enhance password randomness.
   - **Special Characters**: Optional inclusion for further complexity, using symbols like `!@#$%^&*()`.
3. **Command-Line Interface**: With simple command-line options (`-l` for length, `-u` for uppercase, `-n` for numbers, `-s` for special characters), the generator is easy to use, requiring no additional software or GUI dependencies.
4. **Default Settings**: If no options are specified, the generator defaults to a 12-character password with lowercase letters, providing a quick and secure option for casual users.

Together, these features create a flexible and effective tool that simplifies secure password creation, making strong password practices accessible for all Linux users.

4. Implementation

4.1 Tools and Environment

- **Operating System**: Linux
- **Shell**: Bash (Bourne Again Shell)

- **Editor**: Any text editor (e.g., `nano`, `vim`, `gedit`)

4.2 Code Structure

The password generator script is structured to be modular, using functions and variables to handle different aspects of password creation.

1. **Variable Declarations**: Default values are set, such as the password length (12 characters) and character types (lowercase letters only).
2. **Argument Parsing**: Using `getopts`, the script parses user-provided options, allowing them to control the length and character inclusion.
3. **Password Generation Logic**:
    - The `generate_password` function constructs a character set based on user options.
    - A loop iterates over the specified length, appending a randomly selected character to the password string for each iteration.
    - The password is output to the user, fulfilling the specified requirements.

4.3 Script Code

The code for the Linux Password Generator is as follows:

```
bash
Copy code
#!/bin/bash

# Function to generate a random password
generate_password() {
    local length="$1"
    local  charset="abcdefghijklmnopqrstuvwxyz"    #  default:
lowercase letters

    # Include uppercase letters
    if [[ "$include_upper" == "yes" ]]; then
        charset="${charset}ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    fi

    # Include numbers
    if [[ "$include_numbers" == "yes" ]]; then
        charset="${charset}0123456789"
    fi

    # Include special characters
    if [[ "$include_special" == "yes" ]]; then
        charset="${charset}!@#$%^&*()_-+=<>?"
    fi
```

```
 # Generate the password
    password=""
    for _ in $(seq 1 "$length"); do
        index=$((RANDOM % ${#charset}))
        password="${password}${charset:index:1}"
    done

    echo "$password"
}

# Default options
length=12
include_upper="no"
include_numbers="no"
include_special="no"

# Parse command-line options
while getopts "l:u:n:s:" opt; do
    case ${opt} in
        l) length="${OPTARG}" ;;
        u) include_upper="${OPTARG}" ;;
        n) include_numbers="${OPTARG}" ;;
        s) include_special="${OPTARG}" ;;
        *) echo "Invalid option"; exit 1 ;;
    esac
done

# Generate and display the password
echo "Generated Password: $(generate_password "$length")"
```

4.4 Usage Instructions

To generate a password, users can specify their desired length and character inclusion with the following options:

- `-l` specifies the password length.
- `-u` enables uppercase letters.
- `-n` includes numbers.
- `-s` includes special characters.

Example Commands:

- Generate a 16-character password with all options:

  ```
  bash
  Copy code
  ./password_generator.sh -l 16 -u yes -n yes -s yes
  ```

- Generate a simple 12-character password with lowercase only:

```
bash
Copy code
./password_generator.sh
```

5. Testing and Results

The script was tested with various option combinations to verify the functionality:

Test Case	Command	Expected Result
Default (12 lowercase)	`./password_generator.sh`	12-character lowercase password
Length 16 with uppercase and numbers	`./password_generator.sh -l 16 -u yes -n yes`	16-character password with lowercase, uppercase, and numbers
Length 10 with all options	`./password_generator.sh -l 10 -u yes -n yes -s yes`	10-character password with all character types

Each test produced the expected results, demonstrating that the password generator script meets its objectives effectively.

6. Benefits and Limitations

Benefits

The **Linux Password Generator** offers several notable benefits:

1. **Enhanced Security**: By allowing users to create randomized passwords with a mix of uppercase letters, numbers, and special characters, this tool helps protect against brute-force, dictionary, and guessing attacks. Users can generate complex passwords tailored to their security needs, reducing the risk of unauthorized access.
2. **Customizability**: The tool's flexible options for password length and character types allow users to meet diverse requirements, from simple passwords for local applications to highly complex passwords for sensitive accounts or systems. This flexibility makes the tool useful across different security contexts.

3. **Lightweight and Efficient**: Designed using Bash scripting, the generator operates natively in a Linux environment without the need for additional software or dependencies. This ensures that it runs smoothly on any Linux system and can be used across different distributions with minimal setup.
4. **Accessibility for Technical Users**: By offering a simple command-line interface, the tool caters to Linux users who are familiar with terminal commands, making it quick and easy to use.

Limitations

Despite its strengths, the **Linux Password Generator** has some limitations:

1. **Command-Line Usability**: Although the tool is efficient for those comfortable with the command line, it may not be as accessible to non-technical users who prefer graphical interfaces. Integrating a GUI would broaden its appeal and usability.
2. **Pseudo-Randomness**: The generator relies on Bash's `RANDOM` function, which is not cryptographically secure. While sufficient for basic needs, high-security applications might require stronger randomization, such as using `/dev/urandom` or implementing a cryptographic library.
3. **Lack of Storage and Management Features**: The generator does not offer options for securely storing or managing passwords, nor does it integrate with password managers. Users need to save passwords manually, which could introduce potential security risks if done improperly.

7. Future Enhancements

Future improvements could expand the functionality and usability of this project:

1. **Cryptographic Randomness**: Implement a more secure random number generator (e.g., `/dev/urandom`) for better security.
2. **Graphical User Interface**: Develop a GUI version for users unfamiliar with the command line.
3. **Password Strength Indicator**: Add a feature to evaluate the generated password's strength in real time.
4. **Password Storage**: Allow the generated password to be securely saved or encrypted for future use.

5. **Integration with Password Management Tools**: Connect the generator to password managers, enabling users to save or autofill passwords securely.

8. Conclusion

The **Linux Password Generator** project highlights the effectiveness of using Bash scripting in Linux to create a practical, customizable tool for secure password generation. With the increasing prevalence of cyber threats, weak or predictable passwords are a common vulnerability, making the ability to create strong, unique passwords essential. This tool allows users to generate complex passwords by specifying length and selecting character types (lowercase, uppercase, numbers, and special characters), catering to various security requirements.

This password generator is efficient, leveraging Linux's native command-line capabilities, which makes it lightweight and easy to use without the need for third-party software. Through extensive testing, the tool has shown to reliably generate passwords that meet user-defined criteria, fulfilling its objective to enhance password security for Linux users.

Moreover, this project serves as a demonstration of the security benefits of strong password practices and the versatility of Bash scripting for everyday utility development. Looking ahead, potential improvements could include cryptographic randomization for added security, a graphical user interface for broader accessibility, and secure password storage or integration with password managers.

Ultimately, the **Linux Password Generator** provides a valuable resource for promoting safer password habits, empowering Linux users to protect their data and digital identities more effectively.

References

1. **Sobell, M. G.** (2017). *A Practical Guide to Linux Commands, Editors, and Shell Programming*. Addison-Wesley Professional.
    - Comprehensive guide on Linux commands and Bash scripting, essential for understanding the development of command-line tools.

2. **Stallings, W.** (2018). *Cryptography and Network Security: Principles and Practice*. Pearson.
    - Covers critical topics in cryptography, including password security and randomness, explaining why strong, random passwords are essential.

3. **Linux Foundation**. (2021). *Introduction to Bash Scripting*. Retrieved from Linux Foundation
    - An introductory resource on Bash scripting, useful for understanding the basics of creating command-line utilities in Linux.

4. **NIST**. (2020). *Digital Identity Guidelines: Authentication and Lifecycle Management* (NIST SP 800-63B). National Institute of Standards and Technology.
    - Guidelines for secure digital identity management, including best practices for password complexity and strength.

5. **Bash Reference Manual**. (2021). *GNU Bash Reference Manual*. Retrieved from GNU Project
    - The official documentation for Bash, covering commands, syntax, and options relevant to scripting and password generation.