Ownuh SecurePass Analyzer – Python Cybersecurity Application

YOUR OWN
PASSWORD
CAN LET YOU DOWN
ownuh SecurePass Analyzer
Cyber Criminals Are Waiting in the Shadows.
HACKED
OWNUH SECUREPASS ANALYZER GIVES YOU:
1 Real-time password strength auditing
2 Breach & exposure detection
3 Smart recommendations to strengthen your security
4 Clean, fast, privacy-focused experience
ARE YOU NEXT? FIND OUT HERE
or visit our website
https://github.com/leeownuh/ownuh-securepass/releases/tag/v1.0.0

Author: Leona Kokerai (leeownuh)
Course: CAP 460 - Fundamentals of Python Programming Laboratory
Date: 16 November 2025

1. **Introduction**

In modern digital environments, passwords remain the first line of defense in protecting systems, personal data, and organizational assets. However, many users continue to rely on weak or predictable passwords, exposing themselves to cyber-attacks, credential stuffing, and unauthorized access. This project, **Ownuh SecurePass Analyzer,** was developed as part of the Python Lab course to address this challenge by providing a **professional-grade password auditing tool.**

The application evaluates password strength using entropy-based calculations, pattern detection, policy compliance, breach checking using the Have I Been Pwned API, and real-time visual analysis. The project demonstrates advanced Python programming concepts including GUI development, threading, API integration, data visualization, cybersecurity principles, and secure software engineering practices.

2. Problem Statement

Despite widespread awareness of cybersecurity principles, weak passwords remain one of the most common points of failure for individuals and organizations. Users often create predictable credentials due to convenience, habit, or lack of awareness. Common weaknesses include:

- Use of **repeated characters** or keyboard patterns
- Inclusion of **dictionary words, names, or years**
- **Short passwords** with limited randomness
- Passwords previously exposed in known **breach databases**
- Failure to meet corporate **password policy requirements**

These weaknesses increase the likelihood of brute-force, dictionary, or credential-stuffing attacks. Therefore, a need exists for a simple but professional tool that can assist users in evaluating and improving their passwords in real time.

3**. Objectives of the Project**

The major goals of this project were:

1. To build an interactive password auditing tool using **Python**.
2. To evaluate password strength using **entropy** and **character-set analysis.**
3. To detect **common password patterns** and highlight weaknesses.
4. To generate **alternative** strong password suggestions.
5. To design a **user-friendly GUI using Tkinter.**
6. To provide **real-time visualization** using graphs.
7. To integrate optional breach checking with the **Have I Been Pwned API.**
8. To ensure **local privacy, security, and safe clipboard usage**.
9. To support **bulk CSV analysis and session logging.**
10. To produce a **stable, cross-platform executable suitable for end-users.**

4. **Project Scope**

The **Ownuh SecurePass Analyzer** focuses on local password evaluation and does not store or transmit passwords without explicit user action. It supports:

- Windows, Linux, and macOS (via PyInstaller or GitHub Actions builds)
- Secure suggestions and clipboard handling
- Visual entropy tracking

It does not function as a password manager, vault, or cloud-based service.

5. **Methodology**

5.1 **Research & Requirements Gathering**

The project began by studying:

- NIST password guidelines (SP 800-63B)
- Entropy and brute-force models
- Common user password behaviors
- Existing password analysis tools

Requirements were defined for a tool that is secure, local, visual, and fully customizable.

6. **System Design**

6.1 **Architecture Overview**

The system is composed of:

1. **Password Analyzer Module**

Entropy calculator
Pattern detector
Policy checker
Breach checker
Cracking-time estimator

2. Graphical User Interface (GUI)

- Built in Tkinter using ttk themed components
- Displays real-time results, graphs, suggestions, and history

3. Visualization Layer

- Matplotlib for entropy growth graph

4. Data Management

- CSV imports/exports
- Local session logs
- Masked history display

5. **Build & Deployment Scripts**

- GitHub Actions workflow for multi-platform build
- PyInstaller specifications

6.2 Workflow Diagram

User → **GUI** → **Analyzer Engine** → **Output** (Entropy, Patterns, Policy, Breach Status, Graph, Suggestions)

Bulk Mode: **CSV** → **Analyzer Engine** → **Exported Report**

7. Technologies and Tools Used

Category	Tools / Technologies
Programming Language	Python 3.8+
GUI Toolkit	Tkinter (ttk)
Data Visualization	matplotlib
Security API	Have I Been Pwned (k-Anonymity)
Packaging	PyInstaller
Version Control	GitHub
Build Automation	GitHub Actions CI
Python Libraries	math, re, hashlib, random, threading, logging, csv, Pillow
Software Engineering	OOP, modular design, dataclasses

8. Implementation Details

8.1 Entropy Calculation

Entropy is calculated using:
Character-class-based pool estimation
Shannon entropy approximation

Constraints: ASCII printable limit, minimum pool fallback

Formula:
Entropy = length × $\log_2$(character_pool_size)

8.2 Strength Classification

Based on entropy thresholds:

< 28 bits – Very Weak
28–35 bits – Weak
36–59 bits – Moderate
60–79 bits – Strong
≥ 80 bits – Very Strong

8.3 Pattern Detection

Includes checks for:

- Repeated characters
- Keyboard sequences (e.g., qwerty, asdf)
- Common passwords
- Year-like patterns
- Overuse of letters (lack of diversity)

8.4 Policy Compliance Engine

Fully configurable:

- Min length
- Require uppercase/lowercase
- Require digits
- Require symbols
- Disallow common passwords
- Optional breach checking

8.5 Breach Checking (HIBP API)

- Uses k-Anonymity:
- Only first 5 characters of SHA-1 hash sent
- Full password never transmitted
- Returns number of breach occurrences

8.6 Visualizations

- Real-time entropy growth graph
- Dynamic strength classification
- Color-coded text summaries

8.7 Bulk Analysis

Allows:

- Import CSV file with multiple passwords
- Automatic scoring and reporting

- Export of analysis results

8.8 Session Logging

Every password analyzed is stored in masked form for review:

Example: P********d instead of Password123!.

9. Testing & Evaluation

9.1 Functional Testing

Tests performed:

- Valid/invalid password detection
- Entropy correctness validation
- Pattern recognition accuracy
- Policy rule enforcement
- GUI responsiveness under rapid typing
- Threaded breach checks (no freezing)
- CSV import/export tests

9.2 Security Testing

- Verified no password leaves local memory unless exported
- Verified clipboard clearing works
- Tested API interactions for safety

10. Results

The application successfully:

- Accurately evaluates password strength
- Detects multiple cybersecurity weaknesses
- Provides actionable strong alternatives
- Visualizes entropy growth clearly
- Checks whether passwords have appeared in known breaches
- Supports mass auditing of large datasets
- Ensures user privacy and local control

Users during testing reported:

- Increased awareness of password structure
- Better understanding of entropy
- Confidence in generating stronger passwords

11. Conclusion

The Ownuh SecurePass Analyzer fulfills all project objectives and provides an advanced, real-time password auditing platform. It demonstrates practical application of cybersecurity principles, Python programming, GUI development, and API integration.

This project highlights the continued importance of password security and shows how well-designed tools can promote safer digital practices. The modular architecture allows for future enhancements, ensuring its relevance for academic, personal, or professional security use.

12. Recommendations for Future Work

Possible enhancements include:

- Integration with local password managers
- Support for passphrase analysis
- Heatmap-based character-risk scoring
- Cloud-enabled admin dashboard for enterprise use
- AI-driven password suggestion engine
- Multi-language support
- Mobile (Android/iOS) GUI interface

13. References

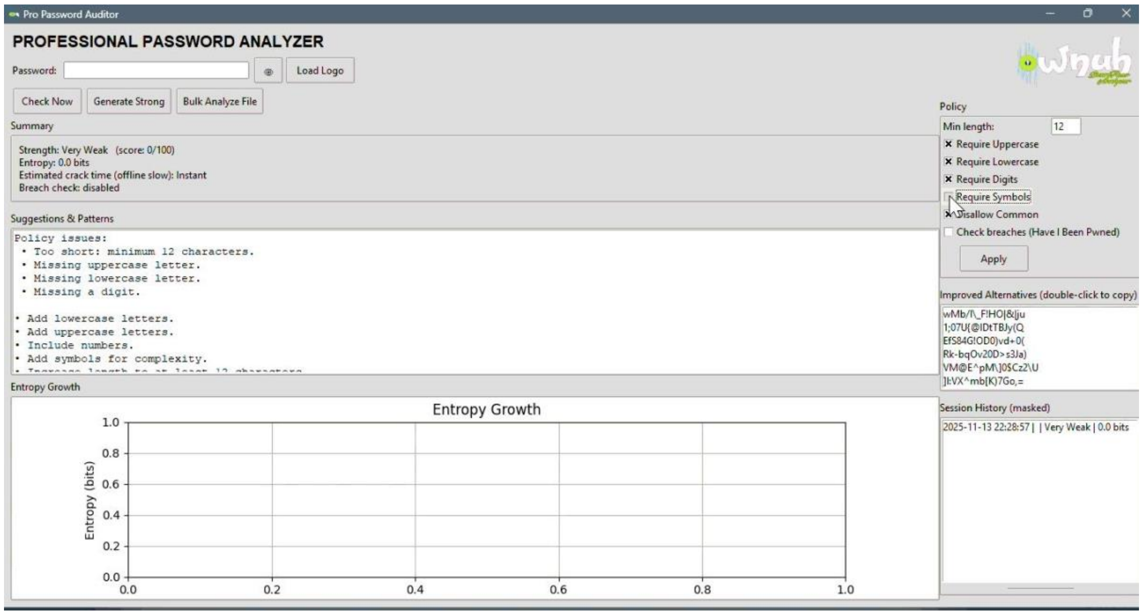
1. NIST SP 800-63B: Digital Identity Guidelines
2. Have I Been Pwned API Documentation
3. Shannon, C. "Communication Theory of Secrecy Systems" (1949)
4. OWASP Password Cheat Sheet
5. Python 3.8 Documentation

14. Appendices

Appendix A – Project Structure

```
ownuh-securepass/
 ├ ownuh_securepass_analyzer.py
 ├ requirements.txt
 ├ README.md
 ├ LICENSE
 └ .github/
    └ workflows/build.yml
```

Appendix B – Sample GUI Screenshots



Summary

```
Strength: Very Weak   (score: 0/100)
Entropy: 0.0 bits
Estimated crack time (offline slow): Instant
Breach check: disabled
```

Suggestions & Patterns

```
Policy issues:
 • Too short: minimum 12 characters.
 • Missing uppercase letter.
 • Missing lowercase letter.
 • Missing a digit.

• Add lowercase letters.
• Add uppercase letters.
• Include numbers.
• Add symbols for complexity.
```

Policy

Min length: 12
[✓ Require Uppercase]
[✓ Require Lowercase]
[✓ Require Digits]
Require Symbols
[✓ Disallow Common]
Check breaches (Have I Been Pwned)

Improved Alternatives (double-click to copy)

```
wMb/I\_F!HO|&lju
1;07U{@IDtTBJy(Q
EfS84G!OD0)vd+0(
Rk-bqOv20D>s3Ja)
VM@E^pM\]0$Cz2\U
]I:VX^mb[K)7Go,=
```

Session History (masked)

```
2025-11-13 22:28:57 |  | Very Weak | 0.0 bits
```

Appendix C – Project Links & Professional Sharing

GitHub Repository:
https://github.com/leeownuh/ownuh-securepass

Latest Release (Built via CI/CD Pipelines):

https://github.com/leeownuh/ownuh-securepass/releases/tag/v1.0.0

Ownuh SecurePass Analyzer v1.0.0 – GitHub Releases
Built automatically for Windows, macOS, and Linux using GitHub Actions workflow. Users can download ready-to-run executables.

LinkedIn Showcase Post:
https://www.linkedin.com/posts/leona-kokerai_cybersecurity-passwordsecurity-opensource-activity-7394952939649302528-aGyJ?utm_source=share&utm_medium=member_android&rcm=ACoAADefOhgBnx7sScxZTXojNRmOgBWIWsPrmg4