ProdigyIfoTech - Security Tools Collection

A collection of Python security and cryptography tools for educational purposes.

Table of Contents

1. Caesar Cipher

File: `caesarCipher.py`

Description

A simple Caesar cipher implementation that shifts letters by a fixed amount. This is an interactive tool that allows users to encrypt and decrypt messages.

Features

- Interactive menu-based interface
- Encrypts text by shifting letters
- Decrypts by reversing the shift
- Preserves case (uppercase stays uppercase, lowercase stays lowercase)
- Non-alphabetic characters remain unchanged

Usage

```
python caesarCipher.py
```

Then follow the menu:

- Option 1: Encrypt a message
- Option 2: Decrypt a message
- Option 3: Exit

Example:

```
```

```
Options
1. Encrypt
2. Decrypt
3. Exit
Enter your choice: (1-3)
1
Enter your text: hello
Enter your shift: 3
Original text: hello
Cipher text: khoor
```

---

2. Image Encryption Tool

File: `image_encryption_tool.py`

Description

Encrypts and decrypts images using pixel manipulation techniques. Supports multiple encryption methods with command-line interface.

Encryption Methods

- **XOR:** Applies XOR operation to pixel values with encryption key
- **SHIFT:** Adds a shift value to each pixel (modulo 256)
- **SWAP:** Reverses RGB channels and shuffles image rows
- **MULTIPLY:** Multiplies pixel values (lossy encryption)

Usage

```
# Encrypt image with XOR method (default)
python image_encryption_tool.py encrypt -i input.jpg -o encrypted.jpg -k 42

# Decrypt with same key
python image_encryption_tool.py decrypt -i encrypted.jpg -o decrypted.jpg -k 42

# Use different method
python image_encryption_tool.py encrypt -i input.jpg -o encrypted.jpg -m shift
-k 100

# View image info
python image_encryption_tool.py info -i image.jpg
```

Parameters

- `-i, --input`: Input image path (required)
- `-o, --output`: Output image path

- `-m, --method`: Encryption method (xor, shift, swap, multiply) - default: xor
- `-k, --key`: Encryption key - default: 42

Supported Image Formats

- JPG, PNG, BMP, GIF, TIFF

---

3. Password Strength Checker

File: `password_strength_checker.py`

Description

Checks password strength based on multiple criteria including length, character types, and composition.

Scoring Criteria

- **Length:** Extra points for passwords longer than 8, 12, and 16 characters
- **Uppercase Letters:** Checks for A-Z
- **Lowercase Letters:** Checks for a-z
- **Digits:** Checks for 0-9
- **Special Characters:** Checks for punctuation marks
- **Character Variety:** Counts unique characters

Current Implementation

The tool analyzes a hardcoded test password and outputs:

- Whether it contains uppercase letters
- Whether it contains lowercase letters
- Whether it contains special characters
- Whether it contains digits
- Overall password length scoring

---

4. Simple Keylogger

File: `keylogger.py`

Description

Monitors keyboard input and logs key presses to a file with timestamps.

⚠ **DISCLAIMER:** For educational and authorized testing only. Unauthorized keylogging is illegal.

Features

- Captures keyboard input in real-time
- Timestamps each key press

- Saves logs to `keyfile.txt`
- Differentiates between character keys and special keys
- Press ESC to exit

Usage

```
python keylogger.py
```

Press `ESC` to stop logging.

Output Format

```
2024-01-03 14:25:35 a
2024-01-03 14:25:36 b
2024-01-03 14:25:37 Key.shift
```

Dependencies

- `pynput` - For keyboard monitoring

---

5. Network Packet Analyzer

File: `packet_sniffer.py`

Description

Captures and analyzes network packets with detailed protocol information. Displays source/destination IPs, ports, protocols, and payload data.

⚠ **DISCLAIMER:** Requires administrator/root privileges. Only use on networks you own or have permission to test.

Captured Information

- **IP Layer:** Source/destination IPs, protocol type, TTL
- **Transport Layer:**
  - TCP: Source port, destination port, sequence/ACK numbers, flags (SYN, ACK, FIN, RST, PSH, URG)
  - UDP: Source port, destination port, length
  - ICMP: Type, code, checksum
- **Data Link:** Source/destination MAC addresses (on Linux)
- **Payload:** Hex and ASCII representation

Usage

```
# Capture packets indefinitely (requires Administrator/root)
python packet_sniffer.py

# Capture specific number of packets
python packet_sniffer.py -c 10

# Specify network interface
python packet_sniffer.py -c 50 -i eth0
```

Parameters

- `-c, --count`: Number of packets to capture (default: 0 = infinite)
- `-i, --interface`: Network interface to sniff on

Requirements

- **Windows:** Run as Administrator
- **Linux/Mac:** Run with `sudo`

Installation

1. Clone the Repository

```
git clone https://github.com/YOUR_USERNAME/ProdigyIfoTech.git
cd ProdigyIfoTech
```

2. Create Virtual Environment

```
# Windows
python -m venv .venv
.venv\Scripts\activate

# Linux/Mac
python3 -m venv .venv
source .venv/bin/activate
```

3. Install Dependencies

```
pip install Pillow numpy pynput
```

Requirements

Python Version: 3.7+

Dependencies:

- `Pillow` - Image processing for encryption tool
- `numpy` - Numerical operations for image encryption
- `pynput` - Keyboard monitoring for keylogger

Optional:

- Administrator/root access for packet sniffer

Project Structure

```
ProdigyIfoTech/
├── caesarCipher.py              # Caesar cipher encryption/decryption
├── image_encryption_tool.py     # Image pixel manipulation encryption
├── password_strength_checker.py # Password strength analysis
├── keylogger.py                 # Keyboard input logger
├── packet_sniffer.py            # Network packet analyzer
├── keyfile.txt                  # Log file for keylogger
├── images/                      # Image directory
│   └── unencrypt1.jpg          # Sample image
└── README.md                    # This file
```

Legal & Ethical Notice

These tools are provided for **educational purposes only**. Ensure you have proper authorization before:

- Running keylogger on any system
- Sniffing network packets
- Testing security tools

Unauthorized access is illegal.

Last Updated: January 3, 2026