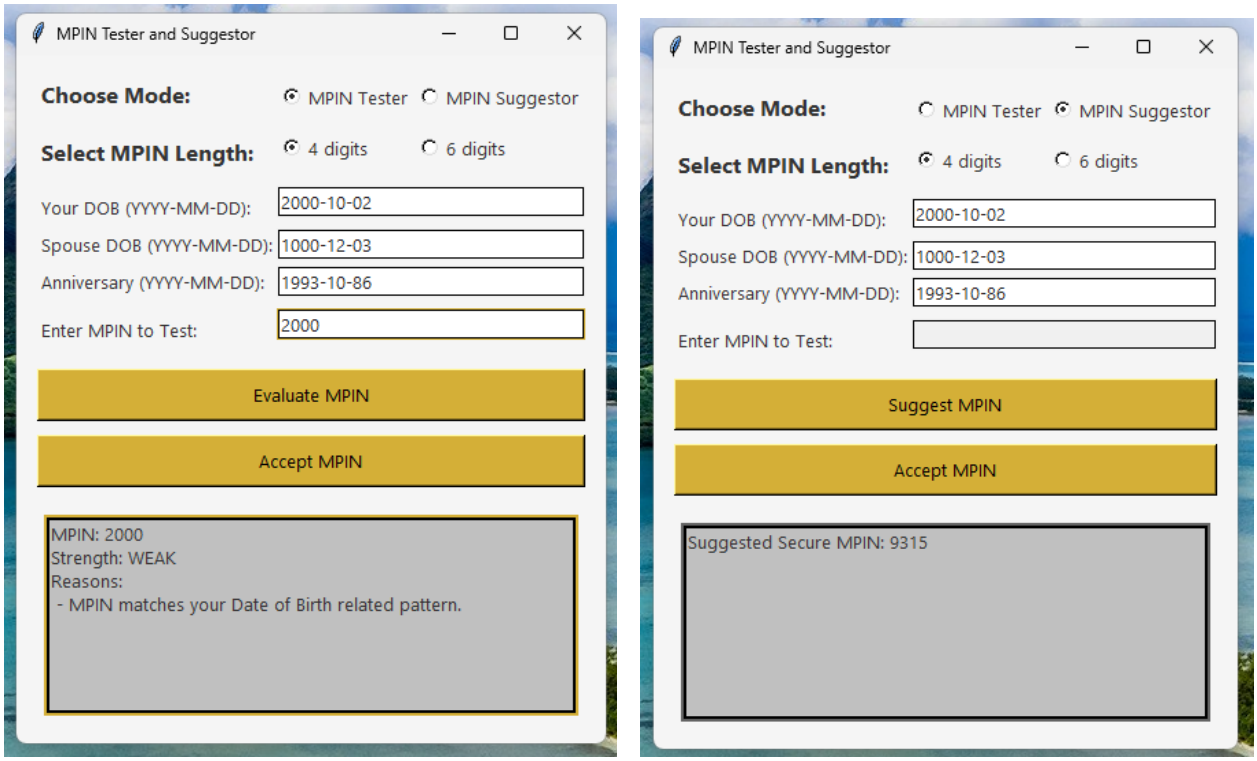
MPIN

This repository contains a basic project to find Weak MPINs by predicting the MPIN as guessable or commonly used one. To fix the Weak MPINS this project suggest the users more secure and unique MPINs for their use using GUI and web application. link



DEMO

MPIN Security Assessment

Download

You can download and use the **MPIN Evaluator and Suggestion Application** for Windows from the link below:

Download MPIN Evaluator and Suggestion Application (.exe)

Introduction: What is MPIN?

MPIN (Mobile Personal Identification Number) is a numeric password used for authentication in mobile banking applications. It generally consists of 4 or 6 digits and serves as a second factor for verifying user identity.

Common Use Cases

- Accessing mobile banking apps (e.g., SBI YONO, HDFC Bank, OneBanc).

- Performing transactions.
- Changing sensitive settings.

⚠ Vulnerabilities of MPINs

1. **Commonly Used PINs**

Users often choose predictable MPINs like:

- `1234`, `0000`, `1111`, `1212`, etc.

These are easily guessable and widely targeted by brute-force attacks.

2. **Demographic-based PINs**

MPINs are frequently derived from personal events, such as:

- **User's Date of Birth**
- **Spouse's Date of Birth**
- **Wedding Anniversary**

Attackers with access to basic personal information may easily guess these MPINs.

🔐 Real-World Countermeasures

- Enforce MPIN complexity rules.
- Prevent use of demographic data in PIN.
- Integrate biometric authentication.
- Implement rate-limiting and lockout mechanisms.

📌 Problem Statement

This project aims to:

1. Detect if the entered MPIN is commonly used.
2. Evaluate if it's related to user demographics (DOB, Spouse DOB, Anniversary).
3. Return `WEAK` or `STRONG` classification.
4. Provide reasons if the MPIN is weak.
5. Support both 4-digit and 6-digit MPINs.
6. Offer a user-friendly GUI and test coverage.

☑ My Solution

🧩 Part A: Detect Commonly Used PINs

Checks the entered MPIN against a list of known common MPINs.

🔍 **Logic**

```
if mpin in commonly_used_pin_list:
    mark_as_weak("COMMONLY_USED")
```

🧩 Part B: Demographic-Based Check

This part checks if the MPIN is derived from any known demographic information that users often use out of convenience. These combinations are predictable and weaken the MPIN's security.

🧠 Inputs

- **User DOB**
- **Spouse DOB**
- **Wedding Anniversary**

🔍 Logic

For each of the above dates, we generate multiple combinations using formats like `DDMM`, `MMDD`, `YYMMDD`, `YYYYMM`, etc., and check whether the given MPIN matches any of these combinations.

```
if mpin in get_date_variants(user_dob):
    mark_as_weak("DEMOGRAPHIC_DOB_SELF")

if mpin in get_date_variants(spouse_dob):
    mark_as_weak("DEMOGRAPHIC_DOB_SPOUSE")

if mpin in get_date_variants(anniversary):
    mark_as_weak("DEMOGRAPHIC_ANNIVERSARY")
```

☑️ Output

- **Strength: WEAK or STRONG**

🧩 Part C: Full Weakness Reasoning System

This part builds upon Part A and Part B by not only determining the **strength** of the MPIN but also clearly **explaining the reasons** behind a weak classification.

🧠 Inputs

- **MPIN** (4-digit)
- **User DOB**
- **Spouse DOB**
- **Wedding Anniversary**

🔍 Logic

We check the MPIN for two types of weaknesses:

1. **Commonly Used MPINs**
   Compared against a predefined list of the most frequently used PINs (e.g., `1234`, `0000`, `1111`, etc.)

2. **Demographic Matches**
   Using the `get_date_variants()` function to generate all meaningful combinations from each date input and checking if the MPIN matches any of those.

Each reason is appended to a list of weakness reasons:

```
weak_reasons = []

if mpin in commonly_used_mpin_list:
    weak_reasons.append("COMMONLY_USED")

if mpin in get_date_variants(user_dob):
    weak_reasons.append("DEMOGRAPHIC_DOB_SELF")

if mpin in get_date_variants(spouse_dob):
    weak_reasons.append("DEMOGRAPHIC_DOB_SPOUSE")

if mpin in get_date_variants(anniversary):
    weak_reasons.append("DEMOGRAPHIC_ANNIVERSARY")

if weak_reasons:
    strength = "WEAK"
else:
    strength = "STRONG"
```

☑ Output

- **Strength: WEAK or STRONG**
- **Reasons (if WEAK):** An array containing any of the following:
- **COMMONLY_USED**
- **DEMOGRAPHIC_DOB_SELF**
- **DEMOGRAPHIC_DOB_SPOUSE**
- **DEMOGRAPHIC_ANNIVERSARY**

If no weaknesses are found, the array will be empty, and the strength will be marked as STRONG.

🧩 Part D: Support for 6-digit MPINs

This part extends the functionality of previous sections to handle **6-digit MPINs**, which are increasingly used for stronger authentication.

🧠 Inputs

- **MPIN** (6-digit)
- **User DOB** (in `YYYY-MM-DD` format)
- **Spouse DOB** (optional)
- **Wedding Anniversary** (optional)

🔍 Logic

The same logic from Part C is reused with minor adjustments to accommodate 6-digit PINs:

1. **Commonly Used 6-digit MPINs**
   A list of frequently used 6-digit MPINs (e.g., `123456`, `000000`, `111111`, etc.) is checked for matches.

2. **Demographic Variants Extended to 6 Digits**
   The `get_date_variants()` function was enhanced to return 6-digit combinations such as:

   - `DDMMYY`, `MMDDYY`, `YYMMDD`, `YYYYMM`, `MMYYYY`, `DDYYYY`, etc.

3. **Reason Identification**
   For every match, the appropriate reason is added to the weakness reasons list.

```
if mpin in commonly_used_6_digit_list:
    reasons.append("COMMONLY_USED")

if mpin in get_date_variants(user_dob):
    reasons.append("DEMOGRAPHIC_DOB_SELF")

if mpin in get_date_variants(spouse_dob):
    reasons.append("DEMOGRAPHIC_DOB_SPOUSE")

if mpin in get_date_variants(anniversary):
    reasons.append("DEMOGRAPHIC_ANNIVERSARY")
```

☑ Output

- **Strength: STRONG or WEAK**
- **Reasons (if weak): List of reasons just like in Part C**

🔄 Backward Compatibility The logic for 4-digit MPINs remains intact, and the application automatically adjusts its checks based on the MPIN length. This enhancement ensures both 4-digit and 6-digit MPINs are evaluated using a consistent and thorough security check mechanism.

🧪 Testing

A set of **20+ test cases** were created to verify all functionalities across Parts A to D.

☑ Test Coverage Includes:

- Commonly used 4-digit MPINs (e.g., `1234`, `1111`)
- Commonly used 6-digit MPINs (e.g., `123456`, `000000`)
- Demographic-based PINs derived from:
  - User's DOB
  - Spouse's DOB
  - Wedding Anniversary
- MPINs that are neither common nor demographic-based (classified as `STRONG`)
- Edge cases like:
  - Empty fields
  - Invalid date formats
  - Leap year dates
  - Same day and month combinations (e.g., `0101`, `1212`)

🩹 Sample Test Case

```
Input:
{
  "mpin": "199802",
  "dob": "1998-02-01",
  "spouse_dob": "1997-04-04",
  "anniversary": "2023-01-01"
}

Expected Output:
{
  "strength": "WEAK",
  "reasons": [
    "DEMOGRAPHIC_DOB_SELF"
  ]
}
```

🛠️ Extra Implementation

To make the MPIN Security Assessment more practical and user-friendly, the following enhancements were added:

💡 MPIN Suggestion System

An optional MPIN Suggestion System was implemented to provide strong alternatives when a weak MPIN is detected.

Key Logic:

- **Randomly generates secure MPINs that do not overlap with user data**
- **Avoids any commonly used MPINs.**
- **Ignores all demographic-related combinations**

Provides:

secure 4-digit and/or 6-digit MPIN suggestions

💻 GUI Application (Tkinter)

A **Tkinter-based graphical user interface (GUI)** was built for non-technical users to easily evaluate their MPIN strength.

Features:

- **Input fields for:**
  - **MPIN (4 or 6 digits)**
  - **User's DOB**
  - **Spouse's DOB**
  - **Wedding Anniversary**
- **"Check Strength" button to process the MPIN**
- **Displays output:**
  - ☑ **MPIN Strength: WEAK or STRONG**
  - 📋 **List of reasons if MPIN is weak**
- **MPIN Suggestion System**

This interface simplifies the evaluation process and helps visualize results clearly and genrate more secure MPINS.

📦 Executable (.exe) File

To make the GUI accessible on machines without Python installed, a **standalone executable file** was created using:

```
pyinstaller --onefile --windowed gui_app.py
```

☑ Summary

This project provides a complete solution to assess the security of MPINs (Mobile Personal Identification Numbers) used in mobile banking apps.

It covers:

- Detection of **commonly used** weak MPINs.
- Identification of **user-specific demographic vulnerabilities**, including:
  - Date of Birth (Self and Spouse)
  - Wedding Anniversary
- Detailed reasoning for weak classification.
- Support for both **4-digit and 6-digit** MPINs.
- A **Tkinter-based GUI application** for user-friendly interaction.
- A compiled **.exe file** for easy distribution without Python setup.

- A **secure MPIN suggestion engine** to guide users toward stronger options.
- A **robust test suite** covering diverse input cases and edge scenarios.

By simulating real-world vulnerabilities and improving MPIN practices, this project helps demonstrate how user behavior and data exposure can compromise mobile banking security — and how it can be mitigated with intelligent checks.
