Project Report: Keylogger Receiver GUI Toolkit

📌 Project Title

Keylogger Receiver GUI Toolkit

🎯 Objectives

- Build a professional, easy-to-use GUI-based tool for receiving keylogger data.
- Support encrypted communication to ensure data confidentiality.
- Provide a built-in EXE generator that embeds the IP address for the sender script.
- Ensure all components are beginner-friendly and ethically aligned.

✨ Features

- 💻 **Hacker-themed GUI**: Dark green-on-black terminal-style interface.
- 🔐 **Secure log reception** using Fernet encryption (symmetric key).
- 🟢 **Start/Stop listener** to receive data in real time.
- 💾 **Save decrypted logs** to text files.
- 📦 **Generate sender EXE** with IP bound inside.
- 🔄 **Switch between interfaces** (Menu, Receiver, Generator).

🛠️ Tools Used & Requirements

Programming Language: Python 3.x

Key Libraries & Modules:

- tkinter: GUI development
- cryptography: Fernet encryption for secure communication
- socket: Handles networking between sender and receiver
- pynput: Logs keystrokes on sender machine
- subprocess: Automates EXE generation via PyInstaller
- pyinstaller: Converts Python files to standalone EXE

Install Requirements:

```
pip install -r requirements.txt
```

Advantages:

- End-to-end encrypted keylogging transmission.
- Portable and can be converted into EXE easily.
- No external database/server setup required.
- Ideal for ethical hacking learning and pentesting practice.

Disadvantages:

- Antivirus software may detect/delete EXE due to keylogging behavior.
- Decryption only works with exact matching Fernet key.
- Requires Python & PyInstaller setup to regenerate EXEs.

📁 Folder Structure

```
keylogger-receiver-gui/
├── receiver_gui.py        # Main GUI with EXE generator + log receiver
├── single_run.pyw         # Sender template with HOST = '<IP>'
├── requirements.txt       # Required dependencies
├── README.md              # Instructions and usage
├── LICENSE                # Legal license for usage
└── dist/                  # Output folder for built EXE files
```

⚙️ Working Explanation

🔐 Encryption & Decryption

- **Fernet** is used to encrypt all keylogs on the sender machine using a shared key.
- The same key is hardcoded into the receiver to decrypt and display logs securely.

📤 Sender Side (Generated EXE):

- Records each keystroke with a timestamp.
- Encrypts it line-by-line using Fernet.
- Sends logs to a given IP via TCP socket every 60 seconds.

📥 Receiver Side (GUI):

- Starts a TCP socket listener.
- Receives encrypted logs.

- Decrypts logs using the hardcoded Fernet key.
- Displays them in a hacker-style terminal on GUI.
- Provides Save option to store logs locally.

🛠 EXE Builder (Inside GUI):

- User inputs IP address.
- Script reads sender template (single_run.pyw).
- Injects IP address into the template (replaces HOST = '<IP>').
- Calls PyInstaller in background to create a .exe from modified file.
- Stores final EXE in /dist folder.

💻 Installation & Usage

1. Install Required Modules

pip install -r requirements.txt

2. Launch Receiver GUI

python receiver_gui.py

3. Build EXE for Sender

- Click 📦 Generate Sender EXE
- Enter your IP → EXE will be created in /dist folder

4. Start Receiving Logs

- Click 📥 Receive Logs
- Press Start to begin listening
- Logs will appear in the text window
- Click Save Logs to store them in a .txt file

🚫 Disclaimer & License

This tool is intended for **learning and ethical use only** — using it on systems without permission is illegal.

👤 Author

Mahavir Harijan
GitHub: @Mahavirharijan