KEYLOGGER WITH ENCRYPTED DATA EXFILTRATION

Saima Shaikh

CYBERSECURITY STUDENT

❖ Introduction:-

- This project demonstrates a Proof of Concept (PoC) for an encrypted keylogger. It securely captures keystrokes using Python, encrypts them with Fernet symmetric encryption, logs them locally with timestamps, and simulates exfiltration to a localhost server. This practical exercise highlights the importance of endpoint security, ethical offensive techniques, and responsible log management practices in the field of cybersecurity.
- The aim is to illustrate how malicious activity can be technically implemented for research purposes, while maintaining ethical practices and safe execution environments. This PoC also includes control mechanisms such as a kill switch for terminating the keylogger safely during demonstrations or testing.

❖ Abstract:-

- The primary objective of this project is to build an ethical keylogger capable of capturing keystrokes in real-time, encrypting the captured data immediately for confidentiality, and simulating secure data exfiltration over a local HTTP server.
- This project includes features such as timestamped encrypted log storage, simulated network data transfer using POST requests, optional startup persistence (for demonstration purposes only), and a kill switch mechanism to terminate execution safely when required. It serves to enhance understanding of both endpoint vulnerabilities and log protection mechanisms while reinforcing responsible cybersecurity practices.

❖ Tools and Technologies Used:-

- **Python 3.13**
- **pynput** — for keyboard event capturing
- **cryptography (Fernet)** — for symmetric encryption
- **requests** — for HTTP POST-based data simulation
- **PyInstaller** — (optional) for converting Python scripts to executables
- **PowerShell / Python HTTP server** — to simulate localhost network server

❖ Steps Involved in Building the Project:-

- 1️⃣ **Install Required Libraries:**

  Install pynput and cryptography libraries using pip.

(Cmd-pip install pynput cryptography)

2️⃣ **Capture Keystrokes using pynput:**

Use Python's pynput library to listen for and record keystrokes.

3️⃣ **Generate and Save an Encryption Key:**

Use cryptography.fernet to generate a symmetric encryption key and store it securely in a file named secret.key.

4️⃣ **Encrypt Captured Keystrokes:**

Encrypt each captured keystroke in real-time using the saved key.

5️⃣ **Store Logs Locally with Timestamps:**

Each encrypted log entry is written to a local file (keylog.enc) alongside its timestamp.

6️⃣ **Simulate Data Exfiltration via HTTP:**

Send encrypted data to a localhost server on port 8000 using HTTP POST requests, simulating a network-based data theft scenario.

7️⃣ **Implement Optional Startup Persistence (Ethical Use Only):**

Optionally copy the executable to the system's startup directory to simulate persistence, only in controlled environments.

8️⃣ **Add a Kill Switch Mechanism:**

Continuously check for the presence of a kill.switch file. If detected, the program terminates safely to avoid uncontrolled execution.

9️⃣ **Decrypt and Review Logs:**

Create a utility to decrypt and display encrypted logs, demonstrating responsible log review practices.

Project Deliverables:-

- keylogger.py — Script that captures, encrypts, logs, and simulates exfiltration
- secret.key — Symmetric encryption key file
- keylog.enc — Encrypted log file containing captured keystrokes
- kill.switch — Optional file to safely stop keylogger execution
- decrypt_log.py — Utility script to decrypt and display logged keystrokes

❖ Conclusion:-

- This encrypted keylogger PoC successfully demonstrates the technical process of capturing and encrypting keystrokes in real-time, securely logging them locally with timestamps, and simulating their transfer to a remote server. It highlights both offensive security techniques for educational and awareness-building purposes and defensive practices such as encryption, log monitoring, and safe execution control.

- The project reinforces key concepts in endpoint security, ethical hacking, and responsible data handling. It provides a practical learning opportunity for cybersecurity students and enthusiasts, showing how malicious activities can be simulated safely within a controlled, consented, and ethical testing environment.