Proof-of-Concept Keylogger with Encrypted Data Exfiltration

Abstract

This report presents the design and implementation of a proof-of-concept (PoC) keylogger that integrates encryption and simulated data exfiltration to a remote server. The primary goal is to understand the techniques used in malware development from a defensive security standpoint. By capturing keystrokes, encrypting logs with symmetric encryption, and mimicking network data transfer, the project highlights how such tools function internally. This research emphasizes the importance of robust endpoint security and responsible cybersecurity education. All work has been conducted in a controlled and ethical environment.

Introduction

Keyloggers are commonly used in cyberattacks to record sensitive user inputs such as passwords and confidential communications. While inherently malicious, studying these tools can help cybersecurity professionals better understand attack vectors and improve defensive strategies. The objective of this project was to build a keylogger that not only captures keystrokes but also applies encryption and simulates exfiltrating the encrypted logs. The final tool includes logging with timestamps, encryption using Python's `cryptography` library, and basic measures for persistence and control. Importantly, this project was created purely for educational and ethical testing purposes within isolated systems.

Tools Used

- **Python**: The main programming language for the project due to its versatility and wide library support.
- **pynput**: A library used to monitor and record keyboard events at a low level.
- **cryptography (Fernet)**: Provides symmetric key encryption, ensuring data confidentiality.
- **base64**: Used to encode the encrypted data for safe storage and transmission.
- **datetime**: Adds precise timestamps to each log entry for chronological tracking.
- **time**: Used to manage time-based events like delays, intervals between log writes, or system checks.
- **Flask (optional)**: Used to simulate a remote server for mimicking data exfiltration via HTTP.

Steps Involved in Building the Project

1. **Environment Setup**:
   Installed required Python libraries using `pip install pynput cryptography flask`.
2. **Keylogger Development**:
   - Utilized `pynput.keyboard.Listener` to track and capture every keystroke.
   - Events were filtered to exclude unnecessary keys like shift or caps lock, ensuring a cleaner log output.
3. **Data Encryption**:
   - A Fernet key was generated and securely stored.
   - Each captured log entry was encrypted using this key before being written to disk.
   - Base64 encoding was applied to ensure file compatibility.
4. **Local Logging**:
   - Logs were saved in a `.log` file with corresponding timestamps.
   - Files were rotated based on size or time interval to simulate realistic logging behavior.
5. **Simulated Data Exfiltration**:
   - A Flask-based mock server was run on localhost.
   - The encrypted logs were "sent" using HTTP POST requests to simulate remote data exfiltration.
6. **Startup Persistence and Kill Switch**:
   - Script optionally added itself to system startup via Windows registry edits or cron jobs on Unix-like systems.
   - A hotkey (e.g., `Ctrl+Shift+Q`) was implemented as a kill switch to terminate the logger safely.

Conclusion

This project successfully demonstrates the core functions of a modern keylogger, including encrypted data handling and simulated exfiltration, within a safe and ethical sandboxed environment. It highlights how threat actors might use relatively simple tools for data theft and the necessity for defensive solutions such as keylogging detection, behavioral monitoring, and endpoint protection.
From an educational standpoint, this project encourages ethical hacking practices and contributes to a deeper understanding of how encryption and persistence can be abused. Future expansions might include stealth enhancements, remote key rotation, or real-time threat detection simulations.