KEYLOGGER WITH ENCRYPTED EXFILTRATION — A CYBERSECURITY POC

Introduction

In an age where data is currency, the act of keystroke logging has evolved into a potent technique both for offensive security researchers and malicious actors. This report presents a Proof of Concept (PoC) implementation of a stealth keylogger that performs encrypted logging and secure exfiltration using Python. The architecture follows a modular approach involving real-time key capture, AES-like encryption using Fernet, periodic flushing, and HTTP-based data transmission to an exfiltration server.

Abstract

This project demonstrates how a Python-based keylogger can be engineered to simulate real-world stealth malware functionalities. The logger captures keystrokes silently, encrypts them using a pre-generated symmetric key, and periodically sends them to a Flask-powered exfiltration server via HTTP POST requests.

All payloads are encrypted before transmission to preserve confidentiality and to evade naive network inspection techniques. The concept also integrates a 'kill switch' mechanism, demonstrating the capability to abort malicious activity gracefully upon detection.

Tools Used

Tool / Module	**Purpose**
pynput	To monitor and log keyboard input
cryptography.fernet	For symmetric encryption of captured logs
Flask	For implementing the exfiltration server
requests	To simulate encrypted POST-based exfiltration
threading, time	For periodic flushing and kill switch polling

Steps Involved in Building the Project

Step 1: Environment Setup

Install Python 3.10+ and necessary packages using pip:

```
pip install pynput cryptography flask requests
```

Step 2: Key Generation

Use Fernet to generate and save a symmetric key:

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
with open("encryption.key", "wb") as f:
    f.write(key)
```

Step 3: Exfiltration Server (exfil_server.py)

A Flask server with an endpoint /exfil listens for encrypted payloads and stores them in received_data.bin.

Step 4: Logger Code (logger.py)

Captures keystrokes, encrypts periodically, and POSTs data to the exfiltration server. Has a kill switch feature.

Step 5: Decryption Utility (decrypt_log.py)

Reads and decrypts .bin files from .syslogs/ using the Fernet key.

Step 6: Testing

Run the exfiltration server, simulate keystrokes using logger.py, then decrypt and verify logs.

Conclusion

This project is a practical demonstration of how basic cybersecurity principles can be both applied and abused. While educational, it serves as a stark reminder of the power that even simple tools wield in the wrong hands. The encrypted keylogger with exfiltration capability highlights the need for endpoint monitoring, behavioral analysis, and proactive security measures. Understanding offensive tooling is key to building better defenses.