Encrypted Keylogger POC (Safe & Ethical Simulation)

Overview

This document describes a safe Proof-of-Concept that simulates the workflow of an encrypted keylogger without capturing real keystrokes. It demonstrates encryption, local log storage, and simulated exfiltration to a localhost server.

Folder Structure

```
encrypted_logger_poc/
  ├── gen_key.py
  ├── fernet_key.txt
  ├── server.py
  ├── client_sim.py
  ├── local_encrypted_logs/
  └── received_logs/
```

Dependencies

Install using:

```
pip install cryptography flask requests
```

Code Files

1. gen_key.py

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
with open('fernet_key.txt', 'wb') as f:
  f.write(key)
print('Fernet key generated and saved to fernet_key.txt')
```

2. server.py

```
from flask import Flask, request, jsonify
from cryptography.fernet import Fernet
from datetime import datetime
import os

with open('fernet_key.txt', 'rb') as f:
  KEY = f.read()
fernet = Fernet(KEY)

app = Flask(__name__)
os.makedirs('received_logs', exist_ok=True)
```

```
@app.route('/upload', methods=['POST'])
def upload():
    data = request.get_json(force=True)
    enc = data['payload'].encode()
    ts = data['timestamp']

    try:
        dec = fernet.decrypt(enc).decode()
    except Exception as e:
        return jsonify({'status': 'error', 'msg': str(e)}), 400

    filename = f'received_logs/log_{datetime.utcnow().timestamp()}.txt'
    with open(filename, 'w') as f:
        f.write(f'Timestamp: {ts}\n')
        f.write(dec)

    print('[SERVER] Received & decrypted:', dec)
    return jsonify({'status': 'ok'})

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=5000)
```

3. client_sim.py

```
import os, json, random, string, requests
from datetime import datetime
from cryptography.fernet import Fernet

with open('fernet_key.txt', 'rb') as f:
    KEY = f.read()
fernet = Fernet(KEY)
os.makedirs('local_encrypted_logs', exist_ok=True)

SERVER_URL = 'http://127.0.0.1:5000/upload'

def fake_keys():
    chars = string.ascii_letters + string.digits + ' '
    return ''.join(random.choices(chars, k=random.randint(5, 30)))

def encrypt(data):
    return fernet.encrypt(data.encode())

def store(enc, ts):
    with open(f'local_encrypted_logs/log_{ts}.enc', 'wb') as f:
        f.write(enc)

def send(enc, ts):
    return requests.post(SERVER_URL, json={'payload': enc.decode(), 'timestamp':
ts}).status_code

if __name__ == '__main__':
    for _ in range(5):
        text = fake_keys()
        ts = datetime.utcnow().isoformat().replace(':', '-')
        enc = encrypt(text)
        store(enc, ts)
        status = send(enc, ts)
        print('[CLIENT] Generated:', text)
        print('[CLIENT] Status:', status)
```