INLIGHN TECH

EXPERIENCE. LEARN. THRIVE

Prerequisites for the Information Stealer Project

By: **Inlighn Tech**

Overview

The Information Stealer project requires students to develop a Python script that extracts sensitive data from a Windows system, including Chrome browser passwords, clipboard content, and system information. The solution uses os for file handling, json and base64 for data parsing, win32crypt and Crypto.Cipher for decryption, sqlite3 for database access, pyperclip for clipboard capture, platform, socket, and uuid for system details, and requests for external IP lookup. To write this script independently, students must master several key concepts and skills from the curriculum. These prerequisites, paired with practical Python code examples, ensure students understand how information stealers work and can implement the script effectively.

Prerequisite Knowledge and Skills

1. **File System and OS Interaction**
   - **What to Know**:

        - Use os.path.join() to construct file paths, os.environ to access environment variables (e.g., USERPROFILE), and os.remove() to delete files.
        - Manage file operations like copying with shutil.copy2().
    - **How It Applies**:
        - The script locates Chrome's Local State and Login Data files, copies the database, and cleans up afterward.
    - **Curriculum Source**:
        - "PYTHON BASICS" Lesson 11 (IO with Basic Files).
        - "WORKING WITH DATA STRUCTURES & FILE HANDLING" Lesson 10 (Opening and Reading Files and Folders Python OS Module).
    - **Practical Prep**:
        - Test file path construction and copying.
    - **Python Code Block**:

```
# File path and copy test
import os
import shutil

user_profile = os.environ["USERPROFILE"]
src_path = os.path.join(user_profile, "Desktop", "test.txt")
dst_path = "test_copy.txt"

with open(src_path, "w") as f:
    f.write("Test file")
shutil.copy2(src_path, dst_path)
print(f"Copied {src_path} to {dst_path}")
if os.path.exists(dst_path):
    os.remove(dst_path)
    print("Cleaned up copy")
```

    -
        - **Run It**: Save as file_test.py, run with python file_test.py (Windows). Prepares for get_decryption_key() and extarct_browser_passwords().

2. **JSON Parsing**
    - **What to Know**:

        - json.loads() parses JSON strings into Python objects; used to extract data from files.
    - **How It Applies**:
        - The script reads Chrome's Local State file to get the encrypted key.
    - **Curriculum Source**:
        - "WORKING WITH DATA STRUCTURES & FILE HANDLING" Lesson 8 (Working with JSON).
    - **Practical Prep**:
        - Test JSON parsing from a file.
    - **Python Code Block**:

```
# Parse JSON from a file
import json

with open("test.json", "w") as f:
   f.write('{"key": "value"}')
with open("test.json", "r") as f:
   data = json.loads(f.read())
   print(f"Parsed data: {data['key']}")
```

    -
        - **Run It**: Save as json_test.py, run with python json_test.py. Prepares for get_decryption_key().

3. **Base64 Decoding**
    - **What to Know**:
        - base64.b64decode() converts base64-encoded strings to binary data.
        - Used to decode encrypted keys or payloads.
    - **How It Applies**:
        - The script decodes Chrome's encrypted key from Local State.
    - **Curriculum Source**:
        - Not explicitly taught; assumes basic Python knowledge (common in security projects).
    - **Practical Prep**:
        - Test base64 decoding.

- **Python Code Block**:

```
# Decode base64 data
import base64

encoded = base64.b64encode(b"secret").decode("utf-8")
print(f"Encoded: {encoded}")
decoded = base64.b64decode(encoded)
print(f"Decoded: {decoded}")
```

-
  - **Run It**: Save as base64_test.py, run with python base64_test.py. Prepares for get_decryption_key().

4. Windows Cryptography with win32crypt

- **What to Know**:
  - win32crypt.CryptUnprotectData() decrypts data protected by Windows DPAPI (Data Protection API).
  - Requires pywin32 library; used for older Chrome passwords or the master key.
- **How It Applies**:
  - The script uses it to decrypt Chrome's master key and older password formats.
- **Curriculum Source**:
  - Not explicitly listed; assumes Windows-specific knowledge (implied in security context).
  - Note: Students need pip install pywin32.
- **Practical Prep**:
  - Test basic decryption (requires a Windows environment).
- **Python Code Block**:

```
# Decrypt with win32crypt (example placeholder)
from win32crypt import CryptUnprotectData

# Simulated encrypted data (actual use requires real DPAPI data)
encrypted = b"dummydata"  # Placeholder
try:
   decrypted = CryptUnprotectData(encrypted, None, None, None, 0)[1]
   print(f"Decrypted: {decrypted}")
```

```
except Exception as e:
    print(f"Error: {e}")
```

-
    - **Run It**: Install pywin32 (pip install pywin32), save as win32crypt_test.py, run with python win32crypt_test.py (Windows). Note: Real testing requires DPAPI-encrypted data, but this introduces the concept for get_decryption_key() and decrypt_password().

5. **AES Decryption with pycryptodome**
    - **What to Know**:
        - Crypto.Cipher.AES decrypts AES-GCM encrypted data using a key, initialization vector (IV), and ciphertext.
        - Chrome uses AES-GCM (v10/v11) for newer passwords.
    - **How It Applies**:
        - The script decrypts Chrome passwords with the key from Local State.
    - **Curriculum Source**:
        - Not explicitly taught; assumes cryptography basics (implied in security projects).
        - Note: Students need pip install pycryptodome.
    - **Practical Prep**:
        - Test AES decryption (simplified example).
    - **Python Code Block**:

```
# AES-GCM decryption example
from Crypto.Cipher import AES

key = b"16bytekey1234567"  # 16-byte key
iv = b"12byteiv1234"       # 12-byte IV
cipher = AES.new(key, AES.MODE_GCM, iv)
encrypted = cipher.encrypt(b"secret")
decipher = AES.new(key, AES.MODE_GCM, iv)
decrypted = decipher.decrypt(encrypted)
print(f"Decrypted: {decrypted}")
```

-
    - **Run It**: Install pycryptodome (pip install pycryptodome), save as aes_test.py, run with python aes_test.py. Prepares for decrypt_password().

6. **SQLite Database Access**
    - **What to Know**:
        - sqlite3.connect() opens a database; cursor.execute() runs SQL queries; fetch results with fetchall().
        - Manage connections with context managers or explicit close().
    - **How It Applies**:
        - The script queries Chrome's Login Data SQLite database for credentials.
    - **Curriculum Source**:
        - Not explicitly listed; assumes basic database knowledge (common in security projects).
    - **Practical Prep**:
        - Test SQLite interaction.
    - **Python Code Block**:

```
# SQLite database test
import sqlite3

conn = sqlite3.connect(":memory:")  # In-memory DB
cursor = conn.cursor()
cursor.execute("CREATE TABLE test (id INTEGER, name TEXT)")
cursor.execute("INSERT INTO test VALUES (1, 'Alice')")
cursor.execute("SELECT * FROM test")
print(f"Data: {cursor.fetchall()}")
conn.close()
```

-
    - **Run It**: Save as sqlite_test.py, run with python sqlite_test.py. Prepares for extarct_browser_passwords().

7. **Clipboard Access with pyperclip**
    - **What to Know**:

        - pyperclip.paste() retrieves the current clipboard content.
        - Useful for stealing copied sensitive data (e.g., passwords).
    - **How It Applies**:
        - The script captures clipboard data with capture_clipboard().
    - **Curriculum Source**:
        - Not explicitly taught; assumes a brief intro (common in automation projects).
        - Note: Students need pip install pyperclip.
    - **Practical Prep**:
        - Test clipboard capture.
    - **Python Code Block**:
    -

```
# Capture clipboard content
import pyperclip

pyperclip.copy("Test clipboard data")
content = pyperclip.paste()
print(f"Clipboard: {content}")
```

    -
        - **Run It**: Install pyperclip (pip install pyperclip), save as clipboard_test.py, run with python clipboard_test.py. Prepares for capture_clipboard().

8. **System Information Gathering**
    - **What to Know**:
        - platform provides OS details (e.g., system(), machine()); socket gets hostname/IP; uuid retrieves MAC address.
        - Parse MAC with regex (re.findall()).
    - **How It Applies**:
        - The script uses these to collect system info in steal_system_info().
    - **Curriculum Source**:
        - "NETWORKING, SOCKETS, AND CYBERSECURITY" PDFs (e.g., "Introduction to Sockets.pdf").
    - **Practical Prep**:
        - Test system info collection.

- **Python Code Block**:

```
# Gather system info
import platform
import socket
import uuid
import re

info = {
   "OS": platform.system(),
   "Hostname": socket.gethostname(),
   "IP": socket.gethostbyname(socket.gethostname()),
   "MAC": ":".join(re.findall("..", "%012x" % uuid.getnode()))
}
print("System Info:", info)
```

-
    - **Run It**: Save as system_info_test.py, run with python system_info_test.py. Prepares for steal_system_info().

9. **HTTP Requests with requests**
    - **What to Know**:
        - requests.get() fetches data from URLs; .json() parses JSON responses.
        - Used for external IP lookup.
    - **How It Applies**:
        - The script queries api.ipify.org for the global IP address.
    - **Curriculum Source**:
        - Not explicitly listed; assumes basic web interaction knowledge.
        - Note: Students need pip install requests.
    - **Practical Prep**:
        - Test an HTTP request.
    - **Python Code Block**:

```
# Fetch external IP
import requests

try:
    response = requests.get("https://api.ipify.org?format=json")
    ip = response.json()["ip"]
    print(f"Global IP: {ip}")
except Exception as e:
    print(f"Error: {e}")
```

- **Run It**: Install requests (pip install requests), save as requests_test.py, run with python requests_test.py. Prepares for steal_system_info().

10. **Exception Handling**
    - **What to Know**:
        - Use try/except to manage file errors, decryption failures, and network issues.
        - Return defaults (e.g., None, empty lists) on failure to keep the script running.
    - **How It Applies**:
        - The script handles errors in key retrieval, decryption, database access, and clipboard/network operations.
    - **Curriculum Source**:
        - "PYTHON BASICS" Lesson 21 (Errors and Exception Handling).
    - **Practical Prep**:
        - Test error handling.
    - **Python Code Block**:

```
# Handle file access errors
import os

try:
    with open("nonexistent.txt", "r") as f:
        print(f.read())
except FileNotFoundError:
    print("File not found")
except Exception as e:
    print(f"Unexpected error: {e}")
```

- ○

    - **Run It**: Save as error_test.py, run with python error_test.py. Prepares for robust error handling throughout the script.

How the Information Stealer Works

- **Concept**:
    - An information stealer extracts sensitive data from a system (e.g., passwords, clipboard, system info) for malicious purposes, simulating real-world malware behavior.
    - This project focuses on educational understanding of such techniques.
- **Workflow**:
    - Extract Chrome passwords by decrypting the Login Data database using the key from Local State.
    - Capture clipboard content with pyperclip.
    - Gather system info (OS, IP, MAC, etc.) using platform, socket, and requests.
    - Print all collected data (could be extended to exfiltrate it).
- **Why Windows-Specific?**:
    - Relies on Windows DPAPI (win32crypt) and Chrome's storage paths, making it platform-specific.

How to Write the Information Stealer Script

Using these prerequisites, students can build the script step-by-step:

1. **Setup**:
    - Import os, json, base64, win32crypt, shutil, sqlite3, Crypto.Cipher, pyperclip, platform, socket, re, uuid, and requests.
2. **Decryption Key**:
    - Write get_decryption_key() to parse Local State and decrypt the key with CryptUnprotectData.

3. **Password Decryption**:
   ○ Implement decrypt_password() to handle AES-GCM (v10/v11) and DPAPI-encrypted passwords.
4. **Browser Passwords**:
   ○ Create extarct_browser_passwords() to copy Login Data, query it with SQLite, and decrypt passwords.
5. **Clipboard**:
   ○ Define capture_clipboard() to grab clipboard content.
6. **System Info**:
   ○ Write steal_system_info() to collect OS, network, and hardware details.
7. **Main Execution**:
   ○ Use if __name__ == "__main__": to call all functions and print results.

Notes for Students

- **Setup**: Install dependencies (pip install pywin32 pycryptodome pyperclip requests). Run on Windows with Chrome installed and saved passwords. Create a test Chrome profile if needed.
- **Engagement**: Run each code block to practice file handling, decryption, and info gathering. Test the full script with python info_stealer.py after ensuring Chrome has saved credentials.
- **Tips**: Use a VM or sandbox for safety; avoid running on a personal machine with sensitive data. Check Chrome's Login Data path exists (%LocalAppData%\Google\Chrome\User Data\Default).

Jump to the Information Stealer Project Description if you are ready now.