UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE
HOUARI-BOUMEDIÈNE
USTHB – ALGER, ALGÉRIE
FACULTÉ D'INFORMATIQUE
DÉPARTEMENT INFORMATIQUE

PROJECT REPORT
Advanced Programming

University of Science and Technology Houari Boumediene

(USTHB)

CHABANE Rym 232331444209
HAMZA Nacera Nour 232331444501
OULID AZOUZ Ahmed Chihabeddine Chafik 232331673903

2025-2026

Table of Contents

1. Introduction

The provided code is a **multi-feature surveillance/monitoring program** designed to run covertly on Windows systems. It functions as spyware that collects various types of sensitive information from an infected computer and exfiltrates it to a remote server. The program operates stealthily by automatically adding itself to Windows startup and running multiple monitoring functions simultaneously.

Primary Functions:

- **Keylogging**: Captures all keystrokes typed by the user
- **Screen capture**: Takes periodic screenshots of the desktop
- **Webcam capture**: Activates the webcam to take photos periodically
- **Wi-Fi credential theft**: Extracts saved Wi-Fi network names and passwords
- **Mouse activity monitoring**: Tracks mouse clicks with application context
- **Active window tracking**: Monitors which applications are being used
- **Data exfiltration**: Sends all collected data to a remote server

2. Extracted Information

The program systematically collects and transmits the following data:

A. System and Identification Data

- **MAC address** of the infected computer
- **Timestamps** for all captured events

B. User Activity Monitoring

- **All typed text** (keylogging) including passwords, messages, emails
- **Mouse click locations** with coordinates
- **Active application names** when clicks occur
- **Periodic screenshots** (every 10 seconds)
- **Webcam photos** (every hour)

C. Network Credentials

- **All saved Wi-Fi network names (SSIDs)**
- **Corresponding Wi-Fi passwords** in plain text
- **Network profiles** stored on the system

D. System Persistence

- **Auto-start registration** in Windows Registry
- **Continuous operation** without user awareness

3. Tools and Libraries Used

Core Libraries:

1. **pynput**: For keyboard and mouse event monitoring
2. **win32gui**: For accessing Windows GUI information (active windows)
3. **requests**: For sending HTTP requests to the remote server
4. **subprocess**: For executing system commands (netsh for Wi-Fi)
5. **cv2 (OpenCV)**: For webcam access and image processing
6. **PIL (Pillow)**: For screenshot capture
7. **winreg**: For Windows Registry manipulation (auto-start)
8. **uuid**: For obtaining the system's MAC address
9. **threading/multiprocessing**: For concurrent execution

System Commands Exploited:

- netsh wlan show profiles: Lists saved Wi-Fi networks
- netsh wlan show profile [NAME] key=clear: Extracts Wi-Fi passwords

Data Transmission:

- **HTTP POST requests** to https://mini-server-90un.onrender.com/send
- **Multipart form data** for both text and file uploads
- **MIME type detection** for proper file transmission

4. Architecture and Execution Flow

Persistence Mechanism:

# *Auto-start implementation*

setup_autolaunch() → Windows Registry →

HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Multi-threaded Architecture:

The program uses three parallel threads for continuous operation:

Thread 1: Webcam Capture (camera())

- Activates default webcam every hour
- Captures and transmits photos
- Cleans up local files after transmission

Thread 2: Screenshot Capture (take_screenshot())

- Captures full screen every 10 seconds
- Transmits images to server
- Deletes local copies

Thread 3: Input Monitoring (start_listeners())

- **Keyboard listener**: Captures all keystrokes, sends text when Enter is pressed
- **Mouse listener**: Records click locations with application context

Main Thread: Wi-Fi Credential Theft

- Extracts saved Wi-Fi credentials on startup
- Sends credentials to server every 15 seconds

Data Flow:

1. **Collection**: Various monitoring functions gather data
2. **Temporary Storage**: Files saved locally in captures/ directory
3. **Transmission**: All data sent to remote server via HTTP
4. **Cleanup**: Local files deleted after successful transmission
5. **Continuous Operation**: All threads run indefinitely

5. Evasion and Stealth Techniques

The code implements several basic but effective evasion techniques designed to maintain covert operation on a compromised system. Primarily, it establishes persistence through Windows Registry manipulation, specifically adding itself to the HKCU\Software\Microsoft\Windows\CurrentVersion\Run key under the benign-sounding name "nice". This ensures automatic execution upon user login without requiring further interaction.

The setup process includes a check for an existing flag file (.autolaunch_setup) to avoid redundant registry modifications that might trigger security alerts. The program operates with no graphical user interface, producing no visible windows or taskbar icons that might alert the user. All local file artifacts are systematically cleaned up after successful transmission to the remote server—screenshots, webcam photos, and other captured data are saved temporarily in a "captures" directory only long enough to transmit, then immediately deleted. Error handling employs silent failure patterns with broad try-except blocks that prevent crashes from being displayed to the user. While the code contains some obfuscation attempts like French-language comments amidst English code and generic function names, it lacks advanced evasion features such as code packing, anti-debugging checks, VM detection, or encryption of its network communications. The malware's fixed timing intervals (10 seconds for screenshots, 15 seconds for Wi-Fi data retransmission, 3600 seconds for webcam photos) and predictable behavior patterns represent vulnerabilities that could facilitate detection by security software monitoring for periodic, automated activities.

6.Conclusion: Technical Merits of the Code

This code demonstrates competent technical implementation across several domains, effectively integrating multiple surveillance functions into a cohesive operational package. The multi-threaded architecture successfully orchestrates simultaneous webcam capture, screenshot recording, and input monitoring without functional interference, showcasing professional-grade concurrency management for continuous operation. Its modular design encapsulates distinct capabilities into dedicated functions, maintaining clean separation of concerns while enabling comprehensive data collection from keyboard inputs, mouse activity, visual captures, and network credentials. The implementation makes sophisticated use of Windows APIs and system utilities, particularly the clever exploitation of netsh commands for Wi-Fi credential extraction and proper Registry manipulation for persistence.

Resource management is handled responsibly with systematic cleanup procedures that minimize disk footprint, and the error handling framework ensures resilient operation through extensive fault-tolerant design. The data pipeline efficiently processes and correlates multiple data streams—typing activity with application context, visual captures with temporal markers—demonstrating advanced surveillance methodology through cross-modality integration. Overall, the code represents a technically proficient execution that successfully achieves holistic system monitoring through the effective coordination of diverse surveillance techniques into a persistent, operational toolkit.