Mục lục

[Danh sách hình vẽ 3](#_Toc452656623)

[Danh sách bảng biểu 4](#_Toc452656624)

[Danh sách từ viết tắt 5](#_Toc452656625)

[MỞ ĐẦU 6](#_Toc452656626)

[TIÊU ĐỀ: Phần mềm gián điệp. 7](#_Toc452656627)

[PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH 1](#_Toc452656628)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 1](#_Toc452656629)

[1. Giới thiệu về hệ điều hành Windows 1](#_Toc452656630)

[2. Giới thiệu về Windows API 1](#_Toc452656631)

[3. Keyboard hooking trong Windows 3](#_Toc452656632)

[4. Chụp ảnh màng hình 4](#_Toc452656633)

[5. Registry và cơ chế tự khởi động cùng với hệ điều hành 4](#_Toc452656634)

[6. Thực thi chương trình bằng System Shell API 7](#_Toc452656635)

[7. Biến môi trường và cách xác định thư mục hệ thống 7](#_Toc452656636)

[8. Đa luồng trong Windows 8](#_Toc452656637)

[9. Script file trong Windows 9](#_Toc452656638)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 10](#_Toc452656639)

[1. Keylogger – ghi lại các phím đã nhấn 10](#_Toc452656640)

[2. Chụp ảnh màng hình 11](#_Toc452656641)

[3. Timer 13](#_Toc452656642)

[4. Chia dữ liệu log vào các file 15](#_Toc452656643)

[5. Tự động chạy với Windows 16](#_Toc452656644)

[6. Tự động cài đặt chương trình khác 17](#_Toc452656645)

[7. Tự động cài đặt bản cập nhật của spy 19](#_Toc452656646)

[8. Cấu trúc của log file 20](#_Toc452656647)

[PHẦN II: LẬP TRÌNH MẠNG 21](#_Toc452656648)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 21](#_Toc452656649)

[1. Tìm hiểu về JDK 21](#_Toc452656650)

[2. Tìm hiểu về TCP Socket 21](#_Toc452656651)

[3. Mô hình client-server 22](#_Toc452656652)

[4. Winsock 25](#_Toc452656653)

[5. Giao thức ARP 25](#_Toc452656654)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 27](#_Toc452656655)

[1. Hoạt động của server 27](#_Toc452656656)

[2. SpyWorker 28](#_Toc452656657)

[3. BdrWorker 29](#_Toc452656658)

[4. Cấu trúc lưu trữ trên server 29](#_Toc452656659)

[5. Cấu trúc gói cài đặt 30](#_Toc452656660)

[6. Quy ước truyền nhận dữ liệu 31](#_Toc452656661)

[7. Sử dụng winsock để truyền nhận dữ liệu với server 31](#_Toc452656662)

[8. Lấy địa chỉ MAC của máy người dùng bằng giao thức ARP 34](#_Toc452656663)

[PHẦN III: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 35](#_Toc452656664)

[CHƯƠNG 1: TRIỂN KHAI 35](#_Toc452656665)

[1. Triển khai client(spy) 35](#_Toc452656666)

[2. Triển khai server 36](#_Toc452656667)

[CHƯƠNG 2: ĐÁNH GIÁ KẾT QUẢ 40](#_Toc452656668)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 41](#_Toc452656669)

[KẾT LUẬN CHUNG 42](#_Toc452656670)

[TÀI LIỆU THAM KHẢO 43](#_Toc452656671)

[PHỤ LỤC 44](#_Toc452656672)

# Danh sách hình vẽ

[Hình 1: Key "Run" cho phép đăng ký chương trình tự động thực thi khi logged-in 7](#_Toc452656090)

[Hình 2: thuật toán xử lý dữ liệu hook 11](#_Toc452656091)

[Hình 3: thuật toán chụp ảnh màng hình 12](#_Toc452656092)

[Hình 4: thuật toán timer 14](#_Toc452656093)

[Hình 5: thuật toán ghi file log 15](#_Toc452656094)

[Hình 6: tự động chạy cùng với hệ thống 16](#_Toc452656095)

[Hình 7: giải nén file zip sử dụng VBScript 18](#_Toc452656096)

[Hình 8: cách thức hoạt động của install.bat 19](#_Toc452656097)

[Hình 9: tự động cập nhật phiên bản mới 20](#_Toc452656098)

[Hình 10: cấu trúc log file 20](#_Toc452656099)

[Hình 11: mô hình client-server 23](#_Toc452656100)

[Hình 12: quá trình làm việc của client và server thông qua socket 24](#_Toc452656101)

[Hình 13: các lớp hoạt động của winsock 25](#_Toc452656102)

[Hình 14: hoạt động của server 27](#_Toc452656103)

[Hình 15: vòng lặp xử lý yêu cầu của Worker 28](#_Toc452656104)

[Hình 16: cấu trúc lưu trữ trên server 30](#_Toc452656105)

[Hình 17: kết nối đến server thông qua winsock 33](#_Toc452656106)

[Hình 18: thư mục mà spy tự động cài vào 35](#_Toc452656107)

[Hình 19: sử dụng task manager để xác định spy có đang chạy không 36](#_Toc452656108)

[Hình 20: sử dụng tiện ích regedit của Windows để xác định spy có đang tự khởi động không 36](#_Toc452656109)

[Hình 21: log được ghi lại tại server nhằm mục đích debug 37](#_Toc452656110)

[Hình 22: log file được ghi lại 37](#_Toc452656111)

[Hình 23: một screenshot file ghi lại ảnh chụp màng hình của người dùng 38](#_Toc452656112)

[Hình 24: đây là log file ghi lại phím bấm của người dùng 38](#_Toc452656113)

[Hình 25: file config của server 39](#_Toc452656114)

# Danh sách bảng biểu

Bảng 1: Các loại value trong registry…………………………………………………12

# Danh sách từ viết tắt

CSDL: cơ sở dữ liệu

Device Context: DC

API: Application programming interface

GUI: Graphic user interface

UI: user interface

SDK: Software development kit

SAM: Security accounts manager

VBS: Visual basic script

COM: Component object model

TCP: Transmission control protocol

IP: Internet protocol

DLL: Dynamic link library

ARP: Address resolution protocol

MAC: Media access control(physical address)

# MỞ ĐẦU

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý Thầy Cô ở Khoa Công nghệ thông tin– Trường Đại học Bách Khoa Đà Nẵng đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này. Nếu không có những lời hướng dẫn, dạy bảo của các thầy cô thì em nghĩ bài báo cáo này của em rất khó có thể hoàn thiện được. Một lần nữa, em xin chân thành cảm ơn thầy. Bài báo cáo Cơ Sở Ngành Mạng thực hiện trong khoảng thời gian gần 4 tháng. Bước đầu đi vào thực tế của em còn hạn chế và còn nhiều bỡ ngỡ. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn học cùng lớp để kiến thức của em trong lĩnh vực này được hoàn thiện hơn.

Em xin gởi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của Khoa Công nghệ thông tin– Trường Đại học Bách Khoa Đà Nẵng đã tạo điều kiện cho em để em có thể hoàn thành tốt bài báo cáo đồ án này. Và em cũng xin chân thành cám ơn cô TS. Phạm Minh Tuấn đã nhiệt tình hướng dẫn hướng dẫn em hoàn thành tốt đồ án này.

Trong quá trình làm đồ án, cũng như là trong quá trình làm bài báo cáo này, khó tránh khỏi sai sót, rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp Thầy, Cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo đồ án sắp tới.

Em xin chân thành cảm ơn!

# TIÊU ĐỀ: Phần mềm gián điệp.

# PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### Giới thiệu về hệ điều hành Windows

Windows là tên của một họ hệ điều hành dựa trên giao diện người dùng đồ hoạ được phát triển và được phân phối bởi Microsoft. Nó bao gồm một vài các dòng hệ điều hành, mỗi trong số đó phục vụ một phần nhất định của ngành công nghiệp máy tính. Các dòng Windows hiện tại gồm Windows NT, Windows Embedded và Windows Phone; chúng có thể bao gồm các phân họ, ví dụ: Windows Embedded Compact (Windows CE) hoặc Windows Server. Các dòng Windows đã bị ngừng gồm Windows 9x và Windows Mobile.

Microsoft giới thiệu một môi trường điều hành có tên là Windows vào 20/11/1985 như một giao diện hệ điều hành đồ hoạ cho MS-DOS để đáp ứng với sự quan tâm ngày càng tăng với các giao diện người dùng đồ hoạ (GUI). Microsoft Windows dần chiếm ưu thế trong thị trường máy tính cá nhân thế giới với hơn 90% thị phần, vượt qua Mac OS, đã được giới thiệu năm 1984. Tuy nhiên, từ 2012, thị phần của nó đã bị tụt lại so với Android, sau đó trở thành hệ điều hành phổ biến nhất năm 2014, khi tính tất cả các nền tảng máy tính mà Windows chạy (giống như Android); vào 2014, số thiết bị Windows đã bán ít hơn 25% của Android.

### Giới thiệu về Windows API

#### Sơ lượt về API

Một giao diện lập trình ứng dụng (tiếng anh Application Programming Interface hay API) là một giao diện mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng. Chẳng hạn, một chương trình máy tính có thể (và thường là phải) dùng các hàm API của hệ điều hành để xin cấp phát bộ nhớ và truy xuất tập tin. Nhiều loại hệ thống và ứng dụng hiện thực API, như các hệ thống đồ họa, cơ sở dữ liệu, mạng, dịch vụ web, và ngay cả một số trò chơi máy tính. Đây là phần mềm hệ thống cung cấp đầy đủ các chức năng và các tài nguyên mà các lập trình viên có thể rút ra từ đó để tạo nên các tính năng giao tiếp người - máy như: các trình đơn kéo xuống, tên lệnh, hộp hội thoại, lệnh bàn phím và các cửa sổ. Một trình ứng dụng có thể sử dụng nó để yêu cầu và thi hành các dịch vụ cấp thấp do hệ điều hành của máy tính thực hiện. Hệ giao tiếp lập trình ứng dụng giúp ích rất nhiều cho người sử dụng vì nó cho phép tiết kiệm được nhiều thời gian tìm hiểu các chương trình mới, do đó khích lệ mọi người dùng nhiều ứng dụng hơn.

Một trong các mục đích chính của một API là cung cấp khả năng truy xuất đến một tập các hàm hay dùng ví dụ như hàm để vẽ các cửa sổ hay các icon trên màn hình. Trong nhiều tình huống, một API thường là một phần của bộ SDK, hay software development kit. Một bộ SDK có thể bao gồm một API cũng như các công cụ/phần cứng, vì thế hai thuật ngữ này không thay thế cho nhau được. Một API tốt thường cung cấp một "hộp đen" hay là một lớp trừu tượng (abstraction layer) bao bọc nó, nhằm đảm bảo là nhà lập trình không thể biết cách hiện thực cụ thể bên trong của mỗi hàm trong API. Điều này làm cho việc thiết kế lại hay cải tiến hàm của API đó trở nên dễ dàng hơn vì nó không làm ảnh hưởng tới các đoạn mã khác mà có sử dụng các hàm đó.

Có hai dòng chính sách đối với việc công bố các APIs:

* Một số công ty bảo vệ APIs của họ một cách mạnh mẽ. Ví dụ, Sony thường chỉ cung cấp API chính thức của PlayStation 2 cho các nhà phát triển PlayStation có đăng kí. Điều này là vì Sony muốn giới hạn những người có thể viết trò chơi trên PlayStation 2, và muốn thu lợi nhuận từ những người này càng nhiều càng tốt. Đây thường là chính sách đối với các công ty mà họ không thu lợi từ việc bán các hiện thực API của họ. Tuy nhiên, PlayStation 3 là công bố hoàn toàn APIs.
* Một số công ty thì cung cấp miễn phí APIs. Ví dụ, Microsoft công bố hầu như hoàn toàn thông tin về các API, để cho các phần mềm có thể được viết chạy trên nền Windows. Việc bán của các phần mềm hãng thứ 3 đồng thời với việc phải mua Hệ điều hành Microsoft Windows. Đây thường là các công ty thu lợi nhuận từ việc bán các hiện thực API.

Một số APIs, chẳng hạn các API là chuẩn cho một hệ điều hành, được hiện thực dưới dạng các thư viện mã độc lập được phân phối kèm theo hệ điều hành. Một số khác thì đòi hỏi nhà sản xuất phần mềm phải tích hợp API trực tiếp vào trong chương trình. Microsoft Windows APIs đi kèm theo hệ điều hành cho phép mọi người có thể sử dụng chúng.

#### Windows API

Bộ Microsoft Windows SDK cung cấp đầy đủ các tài liệu và công cụ cần thiết để xây dựng các chương trình trên nền tản Windows API. Thông thường bộ SDK này sẽ được cài đặt mặt định cùng với công cụ phát triển Visual Studio được phát triển trực tiếp từ Microsoft.

Windows API được phát triển chính dựa trên ngôn ngữ lập trình C, các hàm được trích xuất và các mô tả cấu trúc dữ liệu đều được định nghĩa trong các tài liệu của Microsoft. Các hàm được cung cấp bởi Windows API có thể nhóm thành các loại sau:

* + - Base services: cung cấp các chức năng truy cập vào tài nguyên cơ bản sẵn có của Windows bao gồm: file systems, devices, processes, threads, error handling.
    - Advanced services: cung cấp các hàm truy cập bên ngoài nhân bao gồm: registry, shutdown/restart, start/stop/create Windows Service, manage accounts.
    - Graphics device interface: Cung cấp các hàm làm việc với đồ họa như monitor, printer hoặc các thiết bị khác.
    - User interface: cung cấp các hàm tạo và quản lý các cửa sổ, controls như nút bấm, thanh cuộn...được liên kết với giao diện người dùng.
    - Common dialog box library: cung cấp các hộp thoại chuẩn như hộp thoại mở file, lưu file, chọn màu, font…
    - Common control library: cung cấp khả năng truy cập vào các control đặt biệt của hệ thống như status bar, progress bar, toolbar…
    - Windows shell: cho phép ứng dụng truy cập vào operating system shell.
    - Network services: cho phép ứng dụng truy cập vào các chức năng mạng như NetBIOS, WinSock, RPC…
    - Web: Internet Explorer cũng cung cấp nhiều hàm API làm việc với trình duyệt.
    - Multimedia: cung cấp chức năng chơi file âm thanh, gửi và nhận MIDI messages…
    - Program interaction: cung cấp khả năng tương tác giữa các chương trình trong hệ điều hành Windows với nhau.

Wrapper libraries: ngày nay có nhiều thư viện được phát triển dựa trên Windows API và bao gói các hàm này lại nhằm giảm sực phức tạp trong quá trình làm việc với các API này.

### Keyboard hooking trong Windows

#### Khái niệm về hook

Hook là cơ chế mà một ứng dụng có thể chặn một sự kiện giống như các thông điệp, sự kiện chuột, bàn phím…Hàm chặn một sự kiện cụ thể được gọi là hook procedure, một hook procedure có thể thực thi khi nhận được sự kiện, lúc này nó có thể chỉnh sửa hoặc hủy sự kiện đó.

Hooks thường sẽ làm chậm hệ thống vì nó làm tăng số lượng xử lý cho mỗi thông điệp bị hook. Hook chỉ nên được cài đặt khi cần thiết và gở bỏ khi không cần đến nữa.

Hệ thống hổ trợ nhiều loại hook khác nhau, mỗi loại cung cấp một cơ chế message-handling khác nhau.

Mỗi loại hook sẽ được quản lý trong một hook chain, một hook chain là một danh sách của các con trỏ đặt biệt, các con trỏ này thực chất là hàm callback của hook procedure. Khi một thông điệp xảy ra thì hệ thống sẽ chuyển thông điệp cho lần lượt các hook procedure trong hook chain tương ứng. Các hành động trong hook procedure còn tùy thuộc vào loại hook, một số có thể chỉnh sửa hoặc hủy thông điệp, một số khác chỉ đơn giản là giám sát…

#### Keyboard hooking

Keyboard hooking là một trong những loại hook cơ bản được Windows hổ trợ, nhiệm vụ của nó là chặn bắt phím được nhấn. Keyboard hooking được sử dụng trong chương trình Unikey và một số chương trình tiện ích khác. Đặt biệt, một số kẻ xấu còn lợi dụng công nghệ keyboard hooking để xây dựng chương trình keylogger nhằm đánh cắp thông tin người dùng.

### Chụp ảnh màng hình

#### Device Context(DC)

Các thiết bị độc lập là một trong những phần cốt lỏi của Windows. Ứng dụng có thể vẽ hoặc in ra nhiều loại thiết bị khác nhau, các API hổ trợ làm việc với các thiết bị độc lập này được chứa trong hai thư viện liên kết động. Đầu tiên là Gdi.dll, nó cũng được gọi là giao diện thiết bị đồ họa(GDI), thư viện thứ hai được gọi là trình điều khiển thiết bị. Tên của thư viện thứ hai phụ thuộc vào thiết bị nơi mà ứng dụng của chúng ta sẽ vẽ lên đó.

Ứng dụng phải định nghĩa GDI để nạp lên trình điều khiển thiết bị cụ thể và chỉ cần nạp trình điều khiển thiết bị một lần duy nhất để chuẩn bị cho quá trình vẽ hình ảnh. Những công việc này được tự động thực hiện khi ta tạo một device context(DC). Một DC là một cấu trúc định nghĩa một tập hợp các vật thể đồ họa kèm theo các thuộc tính của nó, chế độ đồ họa cũng như các hiệu ứng output. Ứng dụng của ta sẽ không truy cập thẳng tới DC, thay vào đó chúng ta sẽ làm việc trên cấu trúc gián tiếp bởi việc gọi các hàm khác nhau.

#### Capture desktop

Device context của desktop chứa toàn bộ dữ liệu hình đồ họa ảnh của desktop, ta chỉ cần lấy handle của Desktop Device Context và chuyển đổi nó thành bitmap thông qua các API của hệ điều hành.

### Registry và cơ chế tự khởi động cùng với hệ điều hành

#### Registry trong Windows

Registry là một CSDL phân cấp được sử dụng để lưu trữ các settings cấp thấp của hệ điều hành và các tùy chỉnh của ứng dụng. Nhân, trình điều khiển thiết bị, các dịch vụ, SAM, giao diện người dùng đều có thể sử dụng registry.

Registry chứa hai phần tử cơ bản là key và value. Bạn có thể liên tưởng key như folder và value như file trong hệ thống. Key có thể chứa value hoặc các keys khác giống như mỗi folder chứa file hoặc folder bên trong. Trong registry có các *root key*, chúng như những entry folder. Ví dụ: “HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows” nghĩa là “Windows” được chứa trong key “Microsoft”, key “Microsoft” lại được chứa trong key “Software” và cuối cùng key “Software” được chứa trong *root key* “HKEY\_LOCAL\_MACHINE”. Một số *root key* được định nghĩa trong Win32 API:

* HKEY\_LOCAL\_MACHINE hoặc HKLM
* HKEY\_CURRENT\_CONFIG hoặc HKCC (chỉ có trong Windows 9x và NT)
* HKEY\_CLASSES\_ROOT hoặc HKCR
* HKEY\_CURRENT\_USER hoặc HKCU
* HKEY\_USERS hoặc HKU
* HKEY\_PERFORMANCE\_DATA (chỉ có trong Windows NT, nhưng bị ẩn trong Windows Registry Editor)
* HKEY\_DYN\_DATA (chỉ có trong Windows 9x)

Registry value là tên/dữ liệu được “pair” với key. Mỗi registry value lưu trong registry key có tên duy nhất và không trùng với các value khác.

Mỗi value có một “type” riêng biệt:

|  |  |  |
| --- | --- | --- |
| ID | Tên | Mô tả |
| 0 | REG\_NONE | Lưu bất kỳ dữ liệu gì |
| 1 | REG\_SZ | Lưu một chuỗi string. Chuỗi string được đánh giấu kết thúc bằng giá trị NULL |
| 2 | REG\_EXPAND\_SZ | Lưu chuỗi string có thể mở rộng, thường được dùng để lưu biến môi trường. Chuỗi string được đánh giấu kết thúc bằng giá trị NULL |
| 3 | REG\_BINARY | Lưu dữ liệu nhị phân |
| 4 | REG\_DWORD / REG\_DWORD\_LITTLE\_ENDIAN | Lưu một giá trị nguyên không giấu 32 bits, giấ trị được lưu theo kiểu little-endian |
| 5 | REG\_DWORD\_BIG\_ENDIAN | Lưu một giá trị nguyên không giấu 32 bits, giấ trị được lưu theo kiểu big-endian |
| 6 | REG\_LINK | Liên kết tới bất kỳ registry key khác |
| 7 | REG\_MULTI\_SZ | Chứa danh sách các chuỗi string. |
| 8 | REG\_RESOURCE\_LIST | Danh sách tài nguyên |
| 9 | REG\_FULL\_RESOURCE\_DESCRIPTOR | Mô tả tài nguyên |
| 10 | REG\_RESOURCE\_REQUIREMENTS\_LIST | Danh sách các tài nguyên yêu cầu |
| 11 | REG\_QWORD / REG\_QWORD\_LITTLE\_ENDIAN | Lưu trữ số nguyên 64 bits |

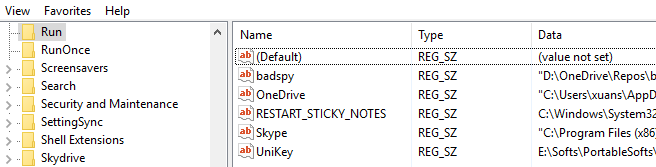
#### *Bảng 1: các loại value trong registry*

#### Root key HKEY\_CURRENT\_USER (HKCU) và cơ chế tự động thực thi chương trình của Windows

*Root key* này lưu trữ những thiết đặt của người dùng hiện tại đang logged-in vào hệ thống. Và như thế mọi thay đổi trong *root key* này sẽ không ảnh hướng tới người dùng khác.

Khi người dùng đăng nhập vào hệ thống, các chương trình đặt biệt sẽ được tự động thực thi ví dụ như: Sticky Notes, Unikey, Antivirus…Có nhiều cách để thống báo cho hệ điều hành tự động thực thi chương trình khi người dùng logged-in trong đó hai cách đơn giản nhất đó là sử dụng folder *Startup* và cấu hình registry. Cách cấu hình registry chuyên nghiệp và “ẩn” được chương trình của ta khỏi tầm mắt của người dùng không chuyên, vì thế đây là sự lựa chọn hoàn hảo và đơn giản cho các ứng dụng độc hại tự động chạy khi người dùng logged-in.

Để cho phép chương trình tự động thực thi khi logged-in bằng registry ta cần thêm một “value” vào registry theo đường dẩn: "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run". Value có tên tùy ý nhưng thường là tên của chương trình, giá trị của value là đường dẩn đến file thực thi của chương trình và kiểu dữ liệu là REG\_SZ.



Hình 1: Key "Run" cho phép đăng ký chương trình tự động thực thi khi logged-in

### Thực thi chương trình bằng System Shell API

Windows cung cấp phương thức thực thi chương trình một cách đơn giản thông qua System Shell API. So với cách truyền thống là sử dụng các hàm API để khởi tạo tiến trình thì Shell API che giấu các bước phức tạp và cung cấp cho chúng ta cách sử dụng đơn giản và trực quan nhất. Các hàm API được sử dụng ở đây là ShellExecute và ShellExecuteEx(cung cấp nhiều tùy chọn hơn), tùy vào mục đích mà ta có thể lựa chọn một trong hai. Chương trình được thực thi từ Shell API có thể chạy ngầm bên dưới hoặc hiển thị cho người dùng thấy.

### Biến môi trường và cách xác định thư mục hệ thống

Biến môi trường là một tập hợp các giá trị động, nó cung cấp các giá trị môi trường cho tiến trình thực thi. Ví dụ như chương trình có thể lấy giá trị của biến TEMP để xác định vị trí thư mục tạm trong hệ thống. Các đường dẩn hệ thống và một số thông tin khác được cài đặt mặt định trong các biến môi trường của Windows. Vì thế ứng dụng có thể xác định chính xác vị trị thư mục hệ thống như System32, AppData, Temp…mà không sợ có sự khác biệt trong các máy tính khác nhau. Dĩ nhiên Windows cung cấp cho ta các hàm API để lấy giá trị của các biến môi trường này một cách dể dàng, một trong những cách đó là sử dụng hàm SHGetFolderPath, nó sẽ lấy giá trị biến môi trường dựa trên một giá trị định danh cho trước được định nghĩa trong các file header.

### Đa luồng trong Windows

#### Tiến trình và tiểu trình

Tiến trình(process) trong Windows tương ứng với một chương trình đang thực thi trong hệ thống, các tiến trình đều chạy độc lập với nhau. Mỗi tiến trình đều có một hoặc nhiều tiểu trình(thread), các tiểu trình của một tiến trình chạy độc lập và gần như song song với nhau. Trong một khoản thời gian chỉ có một tiểu trình được thực thi và việc lựa chọn tiểu trình nào để thực thi phụ thuộc vào bộ lập lịch. Các tiểu trình có mực độ ưu tiên(priority) khác nhau, tiểu trình có mức độ ưu tiên cao sẽ được ưu tiên thực thi hơn tiểu trình có mức độ ưu tiên thấp bởi bộ lập lịch.

Việc sử dụng đa luồng(multi-threading) sẽ tăng tốc độ xử lý trong đa số trường hợp và đồng thời sẽ giúp thực hiện được nhiều công việc cùng lúc hơn.

Trong Windows có hai loại thread đó là foreground thread và background thread.

* Foreground thread: với loại thread này chương trình chính cần phải đợi tất cả foreground thread hoàn tất thì mới có thể kết thúc chương trình.
* Background thread: các thread này sẽ tự động bị ngắt khi chương trình kết thúc.

#### Cách tạo thread trong Windows

Để tạo thread trong hệ điều hành Windows ta có thể sử dụng hàm API mà Windows cung cấp sẵn là CreateThread. Thread mới được tạo ra sẽ tự động được lập lịch để thực thi trên tiến trình hiện tại. CreateThread cần ít nhất một con trỏ tới *thread\_proc*(hàm sẽ được thực thi khi thread chạy), khi mọi xử lý trong *thread\_proc* hoàn tất thì nghĩa là thread đã kết thúc.

#### Đồng bộ trong thread

#### Khi các thread chạy song song với nhau sẽ không tránh khỏi tình trạng xung đột hoặc tranh chấp tài nguyên khi chúng cùng truy cập hoặc chỉnh sửa cùng một tài nguyên nào đó. Ví dụ như nhiều thread cố gắng ghi dữ liệu vào một file thì lúc đó chúng có thể làm mất mát dữ liệu bởi vì một trong số chúng có thể ghi đè dữ liệu hoặc khi một thread đang mở file cùng thời điểm thread khác lại đóng file này.

Để tránh tình trạng đó chúng ta sử dụng khái niệm đồng bộ thread, đảm bảo rằng chỉ có một thread có thể truy cập tài nguyên tại một thời điểm. Các hàm API của Windows cung cấp nhiều phương thức đồng bộ khác nhau như mutex, semaphore, critical section…Trong đó mutex là phương thức đồng bộ đơn giản và hiệu quả nhất. Khái niệm về mutex được ví dụ nhau sau, giả sử có một căn phòng và một chiết chìa khóa, nhiều người cần vào căn phòng đó để làm việc. Mọi người có thể đến cùng lúc nhưng chỉ có người nhanh tay nhất có được chìa khóa, người có chìa khóa sẽ vô được phòng để làm việc và những người khác phải đợi ở ngoài. Khi người trong phòng làm việc xong thì ra ngoài và trả lại chìa khóa về chỗ cũ, lúc này người nhanh tay nhất trong đám người lúc nảy đứng chờ chiếm lấy chìa khóa và vào phòng làm việc, cứ như thế cho đến hết. Ở đây chìa khóa chính là mutex, người chính là thread và căn phòng chính là tài nguyên đang bị các thread tranh chấp. Như đã thấy thì chỉ thread nắm giữ mutex mới có thể làm việc với tài nguyên tranh chấp, các thread khác phải chờ cho đến khi mutex được giải phóng. Cứ như thế lần lược từng thread một làm việc với tài nguyên mà không xảy ra sự tranh chấp nào.

### Script file trong Windows

Windows cung cấp một số loại script file để thực hiện các công việc từ đơn giản đến phức tạp mà không cần tới một trình biên dịch nào. Các script file này sẽ được Windows thực thi từ trên xuống và thực hiện từng lệnh trong file. Chúng ta sẽ tìm hiểu qua hai loại script thông dụng là batch file và visual basic script(VBS) file.

* 1. Bach file

Batch file là một loại script file trong DOS, OS/2 và Windows. Nó bao gồm một tập hợp các lệnh được thực thi bỏi command line và được lưu trữ dưới dạng file text thuần túy. Batch file hổ trợ nhiều lệnh, các cấu trúc điều khiển và cả vòng lặp. Khi batch file thực thi thì một chương trình shell(thường là command.com hoặc là cmd.exe) đọc file và thực thi lần lược các lệnh. Batch file thường có đuôi là .bat, .cmd hoặc .btm. Ta có thể truyền đối số cho batch file từ dòng lệnh, trong batch file để nhận lại các giá trị này ta cần sử dụng các biến đặt biệt(như %0, %1…). Batch file còn hổ trợ việc định nghĩa biến như các ngôn ngữ lập trình, biến trong batch file không có kiểu cụ thể và thường được định nghĩa là biến cục bộ.

* 1. VBS file(VBScript)

Là ngôn ngữ script được phát triển bởi Microsoft dựa trên nền tảng ngôn ngữ lập trình Visual Basic. Nó được thiết kế với tiêu chí là nhẹ, nhanh và đơn giản trên môi trường Windows. VBScript sử dụng COM để truy cập các thành phần của hệ thống, ví dụ như FileSystemObject được sử dụng để làm việc với files. VBScript được cài đặt mặt định trong hệ điều hành Windows.

## CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

### Keylogger – ghi lại các phím đã nhấn

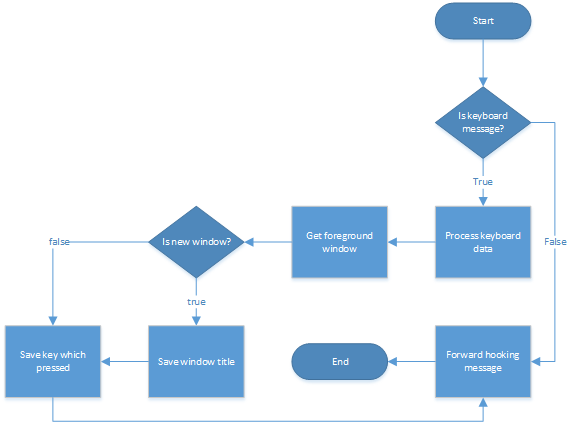
Sử dụng kỉ thuật keyboard hooking để chặn bắt các phím mà người dùng đã nhấn. Khi người dùng nhấn phím bất kỳ thì spy sẽ ghi lại phím được nhấn kèm với tiêu đề của cửa sổ mà người dùng đã focus, như thế ta có thể xác định được nội dung mà người dùng đang nhập là thuộc chương trình nào.

Sử dụng hàm SetWindowsHookEx để đăng ký hook với hệ điều hành, để đăng ký keyboard hooking ta sử dụng cú pháp sau:

SetWindowsHookEx**(**WH\_KEYBOARD\_LL**,** **(**HOOKPROC**)**hook\_proc**,** dll\_instance**,** 0**);**

Trong đó hằng số WH\_KEYBOARD\_LL thông báo với hệ điều hành rằng ta sẽ đăng ký hook bàn phím. Hàm hook\_proc sẽ được gọi khi có bất cứ phím nào được nhấn, khi đó ta chỉ việc phân tích dữ liệu từ phím được nhấn và lưu vào file. Biến số dll\_instance là giá trị handle đến DLL chứa hook procedure, ở đây chính là hook\_proc. Nếu hàm thành công sẽ trả về handle tới hook\_procedure, hàm trả về NULL nếu việc đăng ký hook thất bại.

Sau khi đăng ký hook thành công, việc tiếp theo ta phải làm là xử lý dữ liệu hook khi có phím được nhấn. Thuật toán được mô tả như sau:

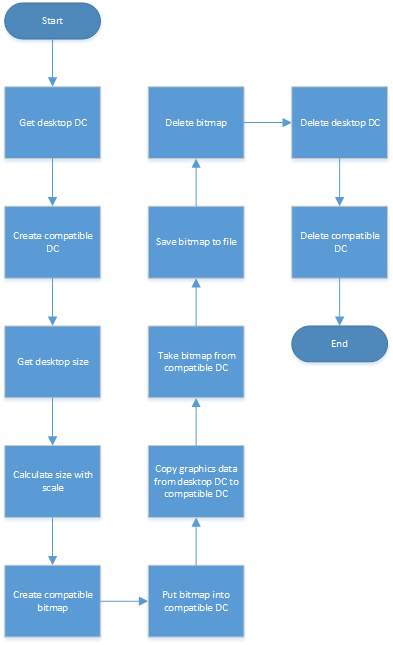


Hình 2: thuật toán xử lý dữ liệu hook

Khi chương trình kết thúc ta cần hủy hook, thật ra việc này là không cần thiết khi đây là một chương trình spy, nghĩa là nó sẽ chạy xuyên suốt từ khi người dùng vô Windows cho đến khi người dùng logoff hoặc shutdown máy. Để hủy hook ta chỉ cần gọi hàm UnhookWindowHookEx với đối số là handle mà hàm SetWindowsHookEx trả về.

### Chụp ảnh màng hình

Spy sẽ tự động chụp ảnh màng hình người dùng định kỳ trong một khoản thời gian nhờ một timer. Ảnh được chụp sẽ được scale nhỏ lại để giảm kích thước file. Để chụp ảnh màng hình, chương trình sử dụng các hàm API của Windows để lấy thông tin đồ họa của Desktop từ đó ta chuyển đổi thành bitmap và lưu vào file. Thuật toán chụp ảnh màng hình như sau:

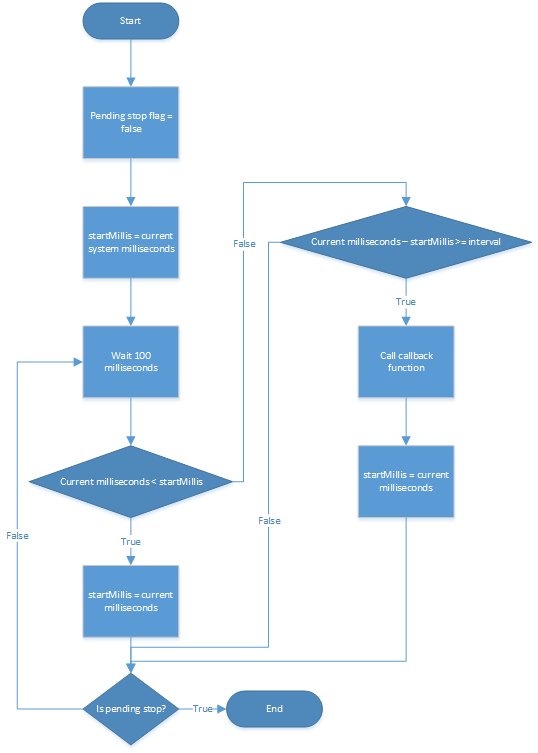


Hình 3: thuật toán chụp ảnh màng hình

Để lấy DC của desktop ta sử dụng hàm API là GetDC với đối số là NULL, sau khi có DC của desktop ta cần tạo một compatible DC bằng hàm CreateCompatibleDC khác tương thích với DC của desktop. Compatible DC có nhiệm vụ làm trung gian để lưu thông tin đồ họa của desktop. Ta cần đặt một bitmap vào bên trong compatible DC, lúc này sử dụng hàm SelectObject, bitmap này sẽ như một tờ giấy để ta có thể “vẽ” hình ảnh của desktop lên nó. Sử dụng hàm StretchBlt để copy hình ảnh trên desktop DC vào trong compatible DC, hình ảnh được copy sẽ “vẽ” thẳng lên trên bitmap. Sau quá trình này ta có được ảnh chụp của desktop được lưu trong bitmap.

### Timer

Nhiệm vụ của nó là thực hiện một công việc sau một khoản thời gian cho trước, sau khi công việc được thực thi thì nó tự động hẹn giờ để thực thi tiếp. Timer chạy trên một thread nền và sử dụng một vòng while để kiểm tra thời gian, nếu thời gian timer đếm lớn hơn hoặc bằng thời gian hẹn giờ thì nó sẽ tự động gọi hàm callback đã cài đặt trước để thực thi công việc. Sơ đồ thuật toán như sau:

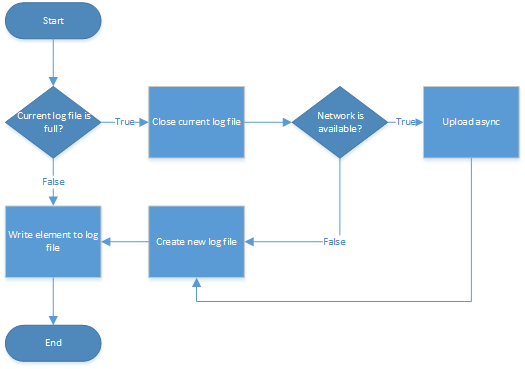


Hình 4: thuật toán timer

Timer sử dụng hàm GetTickCount để lấy số milliseconds hiện tại của hệ thống. Giá trị này sẽ được dùng để tính khoản chênh lệch thời gian từ đó biết được timer đã chờ thời gian mà người dùng cài đặt chưa. Hàm GetTickCount sẽ bị tràn giá trị và bị reset về 0 khi máy người dùng chạy trong khoản thời gian đủ lâu(khoản 49.7 ngày) vì thế để đảm bảo spy luôn hoạt động ổn định trong thời gian lâu ta phải kiểm tra giá trị current tick count so với giá trị ban đầu, nếu giá trị này thấp hơn giá trị ban đầu nghĩa là giá trị đã bị tràn và cần được cập nhật mới.

### Chia dữ liệu log vào các file

Dữ liệu log được tự động ghi vào file, khi dữ liệu này đủ lớn thì chương trình tự động tạo file mới để ghi đồng thời kiểm tra xem nếu máy tính người dùng đang có kết nối mạng thì sẽ tự động gửi các log này lên server bằng một thread nền khác. Sơ đồ thuật toán như sau:



Hình 5: thuật toán ghi file log

Để ghi dữ liệu vào file log cũng như tạo file log mới, chương trình sử dụng các hàm có sẵn của ngôn ngữ lập trình C như fopen, fwrite, fread…Việc upload dữ liệu lên server được thực thi trong một thread nền nhằm giảm sự trì trệ trong việc ghi log và tránh delay phím nhấn của người dùng. Để tạo một thread mới trong Windows ta sử dụng hàm API là CreateThread như sau:

HANDLE thread\_handle **=** CreateThread**(NULL,**

0**,**

**(**LPTHREAD\_START\_ROUTINE**)thread\_proc,**

**NULL,**

0**,**

**&**thread\_id**);**

Phương thức này cần ít nhất một đối số là con trỏ tới hàm thread\_proc, đây là hàm sẽ được thực thi trong thread.

### Tự động chạy với Windows

Spy sẽ tự động chạy khi người dùng đăng nhập vào hệ thống mà không hề hay biết. Để tự động chạy ta chỉ cần ghi một giá trị vào khóa “Run” trong registry của Windows thông qua các hàm API có sẵn.



Hình 6: tự động chạy cùng với hệ thống

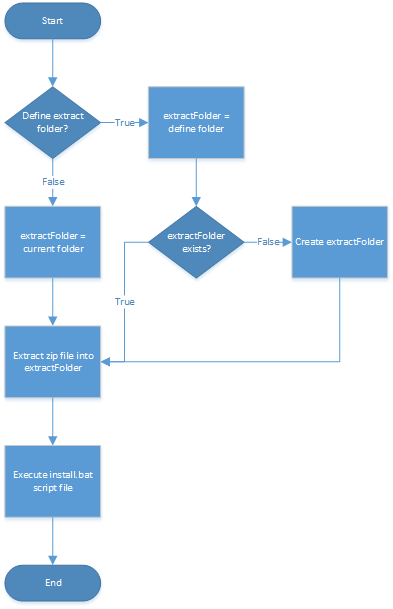
Để làm việc với registry của Windows ta sử dụng hàm RegOpenKeyEx để mở một key xác định, hàm sẽ trả về một handle đến key được mở. Tiếp theo ta sử dụng RegSetValueEx để ghi một value vào key đã mở.

### Tự động cài đặt chương trình khác

Spy có khả năng tự động tải các gói được nén dưới dạng file zip và cài đặt nó vào máy người dùng mà người dùng không hề hay biết. Để tránh việc cài đặt lại các gói đã cài, spy sử dụng một file log để ghi lại các gói đã được cài. Để tự động giải nén và cài đặt ta cần làm hai việc:

#### Giải nén file zip bằng VBScript

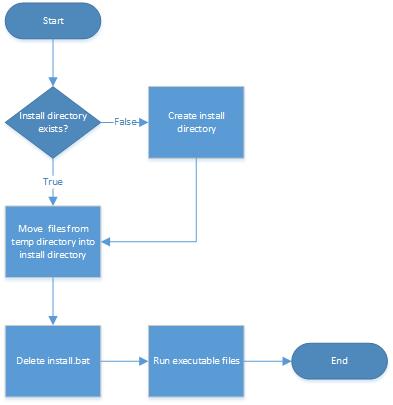
Như đã biết thì VBScript là một trong những script được Windows hổ trợ mặt định, đơn giản mà không kém phần mạnh mẽ. Sau khi script này thực hiện xong công việc thì nó sẽ tự động xóa chính nó vì lúc đó nó không còn cần thiết nữa.



Hình 7: giải nén file zip sử dụng VBScript

#### Thực thi install.bat

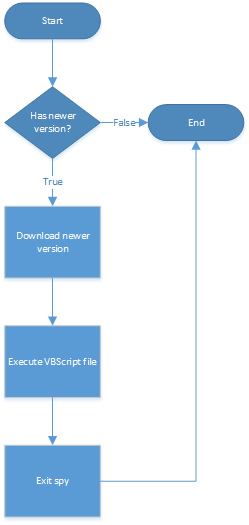
Mỗi gói sẽ có cách thức cài đặt khác nhau, để thuận tiện cho việc thay đổi và chỉnh sửa quá trình cài đặt của các gói này ta sử dụng một batch file. Hai nhiệm vụ cơ bản của file này là di chuyển các files của gói đã giải nén đến thư mục đã định trước trong install.bat, tiếp theo nó sẽ chạy các file thực thi của gói này(cũng đã được định nghĩa trong install.bat). Sau khi batch file này thực thi xong công việc nó sẽ tự động xóa chính nó vì lúc đó nó không còn cần thiết nữa.



Hình 8: cách thức hoạt động của install.bat

### Tự động cài đặt bản cập nhật của spy

Spy cũng có khả năng tự động cài đặt bản cập nhật mới nhất của nó bằng việc xác định phiên bản mới nhất trên server với phiên bản hiện tại, nếu phiên bản hiện tại cũ hơn thì nó sẽ tải về và cài đặt phiên bản mới đó. Khi script cài đặt được thực thi thì spy sẽ tự động thoát để tránh trường hợp file thực thi của spy không thể bị ghi đè trong lúc cài đặt. Cũng tương tự với cơ chế tự động cài đặt và thực thi các gói khác của spy, nó cũng sử dụng đoạn VBScript giải nén và một file batch để cài đặt. Để đơn giản, VBScript sẽ tự động giải nén các file của spy ngay chính thư mục hiện tại của spy, file install.bat chỉ có nhiệm vụ chạy con spy lên.



Hình 9: tự động cập nhật phiên bản mới

### Cấu trúc của log file

Log file là một phần quan trọng của spy, nó cho phép spy ghi lại các thông tin về keyboard hook, screenshot mà còn cả loại thông tin, thời gian ghi…



Hình 10: cấu trúc log file

Dựa vào 2 bytes magic server sẽ xác định được loại file log, 4 bytes tiếp theo sẽ là thời gian mà file log được ghi, các bytes còn lại là nội dung chính của file log.

# PHẦN II: LẬP TRÌNH MẠNG

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### Tìm hiểu về JDK

Bộ công cụ cho người phát triển ứng dụng bằng ngôn ngữ lập trình Java, là một tập hợp những công cụ phần mềm được phát triển bởi Sun Microsystems dành cho các nhà phát triển phần mềm, dùng để viết những applet Java hay những ứng dụng Java - bộ công cụ này được phát hành miễn phí gồm có trình biên dịch, trình thông dịch, trình giúp sửa lỗi (debugger, trình chạy applet và tài liệu nghiên cứu.

Kể từ khi ngôn ngữ Java ra đời, JDK là bộ phát triển phần mềm thông dụng nhất cho Java. Ngày 17 tháng 11 năm 2006, hãng Sun tuyên bố JDK sẽ được phát hành dưới giấy phép GNU General Public License (GPL), JDK trở thành phần mềm tự do. Việc này đã được thực hiện phần lớn ngày 8 tháng 5 năm 2007 và mã nguồn được đóng góp cho OpenJDK.

### Tìm hiểu về TCP Socket

#### Giao thức TCP

Giao thức TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet, trong đó có WWW, thư điện tử và file transfer.

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin không đáng tin cậy. TCP làm nhiệm vụ của tầng giao vận trong mô hình OSI đơn giản của các mạng máy tính.

Các ứng dụng gửi các dòng gồm các byte 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các đoạn (segment) có kích thước thích hợp (thường được quyết định dựa theo kích thước của đơn vị truyền dẫn tối đa (MTU) của tầng liên kết dữ liệu của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một liên mạng tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (sequence number). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin báo nhận" (acknowledgement) cho các gói tin đã nhận được thành công; một "đồng hồ" (timer) tại nơi gửi sẽ báo time-out nếu không nhận được tin báo nhận trong khoảng thời gian bằng một round-trip time (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (giá trị kiểm tra) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

#### Network socket

Một network socket là một đầu cuối của một kết nối thông qua mạng máy tính. Socket là một handle cho phép một chương trình cục bộ có thể sử dụng connection thông qua networking API. Ví dụ, để gửi “Hello, world” thông qua TCP tới port 80 của host có địa chỉ là 1.2.3.4 ta làm lần lược 4 bước như sau:

Socket socket **=** getSocket**(**type **=** "TCP"**)**

connect**(**socket**,** address **=** "1.2.3.4"**,** port **=** "80"**)**

send**(**socket**,** "Hello, world"**)**

close**(**socket**)**

Đầu tiên ta tạo một socket và kết nối nó đến host 1.2.3.4 ở port 80 theo giao thức TCP, Sau khi socket được tạo ta có thể gửi chuỗi “Hello, world” thông qua socket đó. Sau khi làm việc xong với socket ta cần đóng lại để giải phóng tài nguyên.

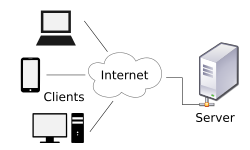
Socket API là một API thường được cung cấp bởi hệ điều hành, nó cho phép ứng dụng có thể điều khiển và sử dụng network socket.

Một số loại internet socket sẵn có như:

* Datagram sockets: cũng được biết như là một socket không kết nối, nó sử dụng giao thức UDP.
* Stream sockets: cũng được biết như là một socket có kết nối và sử dụng giao thức TCP.
* Raw sockets: thường sẵn có ở routers và những thiết bị mạng khác. Ở đây lớp transport bị bỏ qua và ứng dụng có thể quy cập vào header của gói tin.

### Mô hình client-server

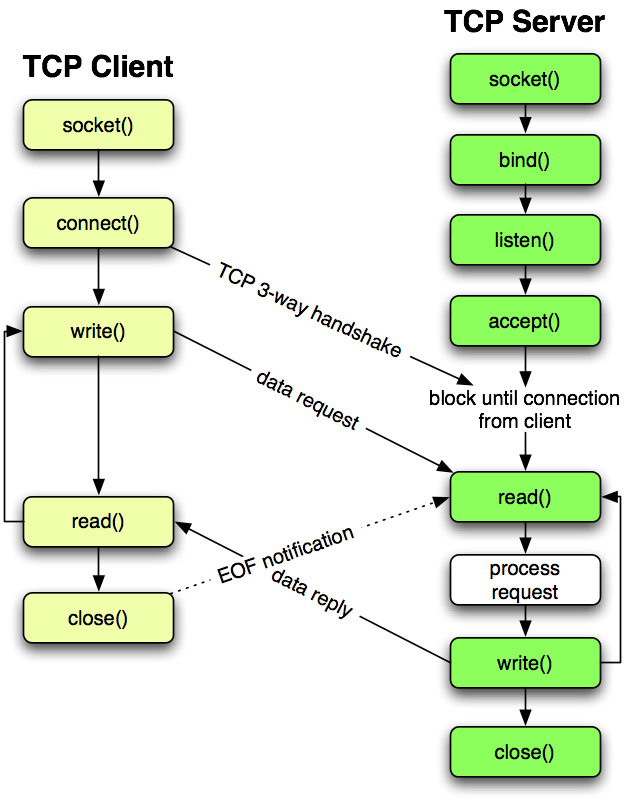
Đây là một mô hình nổi tiếng trong mạng máy tính, được áp dụng khá rộng rãi và cũng là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con(đóng vai trò là máy khách) gửi một yêu cầu (request) để máy chủ (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.



Hình 11: mô hình client-server

Thuật ngữ server được dùng cho những chương trình thi hành như một dịch vụ trên toàn mạng. Các chương trình server này chấp nhận tất cả các yêu cầu hợp lệ đến từ mọi nơi trên mạng, sau đó nó thi hành dịch vụ và trả kết quả về máy yêu cầu. Một chương trình được coi là client khi nó gửi các yêu cầu tới máy có chương trình server và chờ đợi câu trả lời từ server. Để một chương trình server và một chương trình client có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn để nói chuyện, chuẩn này được gọi là giao thức. Nếu một chương trình client nào đó muốn yêu cầu lấy thông tin từ server thì nó phải tuân theo giao thức mà server đó đưa ra. Bản thân chúng ta khi cần xây dựng một mô hình client/server cụ thể thì ta cũng có thể tự tạo ra một giao thức riêng nhưng thường chúng ta chỉ làm được điều này ở tầng ứng dụng của mạng. Với sự phát triển mạng như hiện này thì có rất nhiều giao thức chuẩn trên mạng ra đời nhằm đáp ứng nhu cầu phát triển này. Các giao thức chuẩn (ở tầng mạng và vận chuyển) được sử dụng rộng rãi nhất hiện nay như: giao thức TCP/IP, giao thức SNA của IBM, OSI, ISDN, X.25 hoặc giao thức LAN-to-LAN NetBIOS. Một máy tính chứa chương trình server được coi là một máy chủ hay máy phục vụ (server) và máy chứa chương trình client được coi là máy tớ (client). Mô hình mạng trên đó có các máy chủ và máy tớ giao tiếp với nhau theo 1 hoặc nhiều dịch vụ được gọi là mô hình client/server. Thực tế thì mô hình client/server là sự mở rộng tự nhiên và tiện lợi cho việc truyền thông liên tiến trình trên các máy tính cá nhân. Mô hình này cho phép xây dựng các chương trình client/server một cách dễ dàng và sử dụng chúng để liên tác với nhau để đạt hiệu quả hơn.

Trong mô hình client-server, ứng dụng có thể sử dụng socket để truyền nhận dữ liệu, quá trình làm việc của TCP Server và TCP Client như sau:



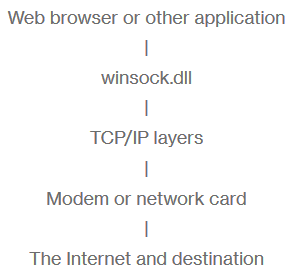
Hình 12: quá trình làm việc của client và server thông qua socket

Về phía server, chúng cần tạo ra một socket và bind đến mộ endpoint(bao gồm một địa chỉ IP và một port trên server), tiếp theo nó sẽ lắng nghe các kết nối tại endpoint đó. Khi một client kết nối đến, nó sẽ thực hiện quá trình truyền nhận dữ liệu với client và sau đó kết thúc bằng cách đóng socket.

Về phía client, nó cũng tạo ra một socket nhưng thay vì bind và lắng nghe tại một endpoint trên máy client thì nó cần kết nối đến một endpoint trên server. Sau khi kết nối thì nó sẽ gửi nhận dữ liệu với server và cuối cùng đóng socket để kết thúc.

### Winsock

Windows Sockets API hay còn gọi ngắn gọn là Winsock, nó là một công nghệ đặt biệt định nghĩa cách mà các ứng dụng mạng trong Windows có thể truy cập tới các dịch vụ mạng, thường là TCP/IP. Windows Sockets code và thiết kế dựa trên BSD sockets nhưng nó cung cấp thêm các chức năng để cho phép API tuân theo chuẩn lập trình của Windows. Winsock chạy giữa ứng dụng và TCP/IP, các yêu cầu trước truyền theo thứ tự như sau:



Hình 13: các lớp hoạt động của winsock

Winsock cung cấp một giao diện chung cho các phiên bản khác nhau của Windows. Kể từ Windows 95, winsock đã trở thành một phần của hệ điều hành Windows nhưng với các phiên bản cũ hơn thì người ta phải cài đặt nó. Winsock chạy như một thư viện động(DLL file), nghĩa là nó chỉ được nạp vào bộ nhớ khi một ứng dụng cần nó.

### Giao thức ARP

Tại tầng network của mô hình OSI, chúng ta thường sử dụng loại địa chỉ IP, các địa chỉ này phân thành hai phần riêng biệt là địa chỉ mạng và địa chỉ máy. Cách đánh số như vậy nhằm giúp việc tìm ra các đường kết nối từ hệ thống mạng này sang hệ thống mạng khác được dễ dàng hơn. Các địa chỉ này có thể thay đổi được tùy vào ý người dùng.

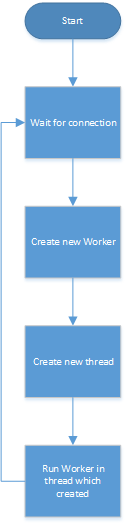
Trên thực tế card mạng chỉ có thể kết nối với nhau theo địa chỉ MAC, địa chỉ cố định và duy nhất của phần cứng do đó ta phải có một cơ chế để chuyển đổi qua lại giữa các địa chỉ này với nhau.

Giao thức ARP ra đời để giải quyết vấn đề đó, nó cung cấp khả năng chuyển đổi từ địa chỉ IP sang địa chỉ MAC. Về nguyên tắt hoạt động, khi một thiết bị muốn biết địa chỉ MAC của một thiết bị mạng nào đó mà nó đã biết địa chỉ IP, nó sẽ cần gửi một ARP request bao gồm địa chỉ MAC của nó và địa chỉ IP của thiết bị mà nó cần biết địa chỉ MAC. Yêu cầu này sẽ được broadcast ra toàn bộ mạng, và các thiết bị nhận được yêu cầu này sẽ so sánh địa chỉ IP trong ARP với IP của nó, nếu bằng thì nó sẽ phản hồi lại địa chỉ MAC của nó.

## CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

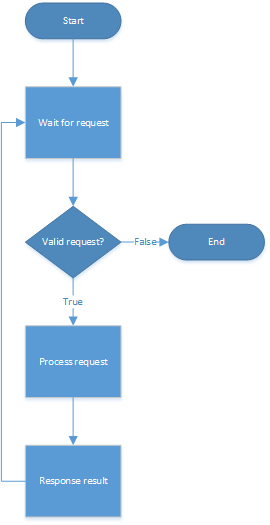
### Hoạt động của server

Server sẽ được chia ra làm 2 phần riêng biệt, một phần sẽ có nhiệm vụ nhận log từ spy, một phần có nhiệm vụ truyền các gói cài đặt cho spy nếu có yêu cầu. Việc chia làm hai phần như thế này sẽ làm giảm độ phức tạp trong xử lý. Server sẽ sử dụng lớp ServerSocket có sẵn của java để khởi lắng nghe kết nối, khi một client kết nối đến nó sẽ sinh ra một lớp Worker làm nhiệm vụ giao tiếp với client đó, mọi hoạt động của Worker sẽ diễn ra trong một thread mới tách biệt hoàn toàn với thread mà server đang chạy. Khi một Worker được sinh ra và khởi chạy thì server sẽ không còn cần quan tâm đến nó nữa, nhiệm vụ bây giờ của server là quay trở lại lắng nghe kết nối của client tiếp theo.



Hình 14: hoạt động của server

Khi Worker thực thi, nó sẽ nhận các yêu cầu từ client, phân tích các yêu cầu đó và phản hồi kết quả về cho client.

****

Hình 15: vòng lặp xử lý yêu cầu của Worker

Mỗi server sẽ sinh ra một loại Worker cụ thể để xử lý các yêu cầu chuyên biệt của server đó, chúng ta sẽ tìm hiểu về hai loại Worker tương ứng với hai server ở phần tiếp theo.

### SpyWorker

Nhiệm vụ chính của nó là nhận các thông tin về máy của người dùng và log file. Ở phiên bản hiện tại, SpyWorker yêu cầu nhận các thông tin như:

* Log file: là dữ liệu chính mà spy thu thập được bao gồm hai loại log là keyboard log và screenshot log. Các loại log file này được phân biệt dựa vào 2 bytes đầu tiên của file mà client gửi lên.
* MAC: server sẽ phân biệt các máy của victim thông qua giá trị địa chỉ MAC này.
* Hostname: tên host của máy victim, giá trị này sẽ được sử dụng để xác định một cách tương đối các victim.
* Version: đây là phiên bản của spy, mỗi phiên bản của spy sẽ có thể có cấu trúc lưu trữ khác nhau. Dữ liệu sẽ được lưu trữ theo từng loại phiên bản của spy, chúng ta sẽ tìm hiểu cụ thể hơn ở phần tiếp theo.

### BdrWorker

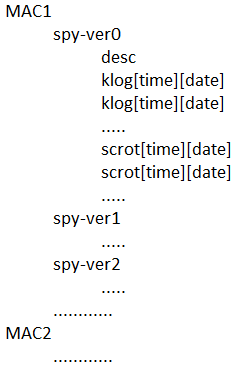
Nhiệm vụ của nó là nhận và xử lý các yêu cầu liên quan đến cập nhật cũng như thông tin các gói cài đặt khác mà server muốn spy cài vào máy victim.

Với phiên bản hiện tại BdrWorker hổ trợ 3 tính năng:

* Kiểm tra cập nhật của spy: spy cần gửi thông tin phiên bản của nó lên server, tiếp theo server sẽ kiểm tra xem phiên bản spy mới nhất mà server có là bao nhiêu và so sánh với phiên bàn mà client gửi lên. Sau khi phân tích dữ liệu thì BdrWorker sẽ phản hồi lại giá trị yes/no tương ứng với có phiên bản mới nhất hay không.
* Lấy danh sách toàn bộ các gói sẵn có trên server: với yêu cầu này, server sẽ quét toàn bộ các gói(mỗi gói tương ứng với một chương trình mà server muốn spy cài đặt) sẵn có trên server và phải hồi lại danh sách các gói này. Client dựa vào danh sách này và so khớp với các ID của các gói đã tải trước đó, nếu có gói nào mới thì nó sẽ tải và cài đặt gói đó dựa vào ID này.
* Tải gói: client sẽ cần phải gửi ID của gói cần tải lên server, lúc này server sẽ kiểm tra gói đó có thực sự tồn tại. Nếu gói đó tồn tại server sẽ gửi dữ liệu của gói đó về cho client. Sau khi tải xong dữ liệu của gói, client có thể giải nén và tiến hành cài đặt vào máy victim.

### Cấu trúc lưu trữ trên server

Server sẽ lưu trữ các thông tin của victim theo cấu trúc cây thư mục, mỗi victim sẽ được cung cấp riêng một thư mục dựa vào địa chỉ MAC như sau:



Hình 16: cấu trúc lưu trữ trên server

Dựa vào MAC ta sẽ phân biệt được các máy của victim với nhau, tiếp theo spy-version folder sẽ tương ứng với phiên bản của spy đã gửi log. Vì các spy có thể khác phiên bản nhau, như thế cấu trúc lưu trữ cũng có thể sẽ khác nhau, cho nên việc phân tách các folder riêng biệt cho các phiên bản là cần thiết.

Với phiên bản hiện tại, trong mỗi folder của spy sẽ chứa 2 loại file là desc file và log file:

* Desc file(description file): đây là file chứa thông tin mô tả về victim, các thông tin này hữu ích để phân biệt các victim ngoài địa chỉ MAC. Hiện tại mới hổ trợ lưu trữ hostname của victim.
* Log file: các file này sẽ chứa nội dung chính của spy gửi lên, định dạng của file này gồm 2 phần là prefix(loại log) và postfix(thời gian log được tạo ở client).
  + Klog file(keyboard log file): file lưu trữ dữ liệu hook bàn phím của victim.
  + Scrot file(screenshot log file): file lưu trữ dữ liệu chụp ảnh màng hình của victim.

### Cấu trúc gói cài đặt

Server lưu trữ các gói cài đặt của các chương trình khác hoặc chính nó trong file zip. Tên của file này sẽ mô tả đầy đủ nội dung mà nó chứa như ID của gói, tên gói và phiên bản của gói. Format như sau: id{name}.x.y.z trong đó:

* ID: là ID định danh của gói, mỗi gói chỉ có một ID duy nhất, với phiên bản hiện tại hổ trợ ID từ 0 đến 255. ID 0 là ID đặt biệt, nó được sử dụng để lưu trữ gói cài đặt của chính spy.
* Name: tên của gói, không có khoản trống.
* X.y.z: phiên bản của gói theo định dạng major.minor.revision.

### Quy ước truyền nhận dữ liệu

Client và server sẽ quy ước cách thức giao tiếp với nhau như sau: Yêu cầu của client sẽ gồm 2 phần là loại yêu cầu(contentType) 1 byte và kích thước nội dung(contentLength) 4 bytes.

* ContentType: loại yêu cầu để phân biệt các yêu cầu với nhau, từ đó server biết phải xử lý và phản hồi những gì cho client.
* ContentLength: nếu giá trí này lớn hơn 0 nghĩa là client sẽ gửi thêm dữ liệu “extra” cho server và kích thước của dữ liệu này sẽ là giá trị của contentLength.

Các contentType hiện tại được hổ trợ là:

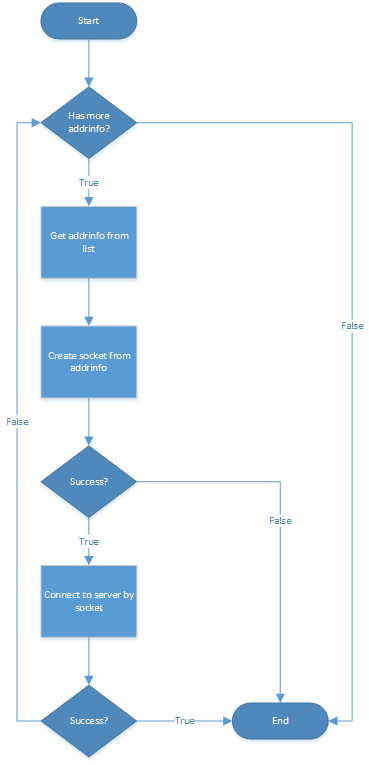
* HEADER\_FILE: thông báo cho server biết client sẽ gửi log file.
* HEADER\_VERSION: client gửi phiên bản của spy lên server.
* HEADER\_MAC: client gửi địa chỉ MAC của victim lên server.
* HEADER\_HOSTNAME: client gửi hostname của victim lên server.
* HEADER\_BDR\_CHK\_UPDATE: client kiểm tra cập nhật của spy.
* HEADER\_BDR\_DWN\_PACK: client yêu cầu tải gói dựa vào ID.
* HEADER\_BDR\_DT\_FILENAME: server gửi tên gói cho client.
* HEADER\_BDR\_DT\_FILESIZE: server gửi thông tin kích thước gói cho client.
* HEADER\_BDR\_DT\_FILECHUNK: server gửi một đoạn dữ liệu của gói cho client.
* HEADER\_BDR\_PACKS\_LIST: client muốn lấy danh sách các gói có trên server.

### Sử dụng winsock để truyền nhận dữ liệu với server

Như đã tìm hiểu, winsock bộ API được Windows cung cấp để các ứng dụng có thể làm việc với mạng. Để sử dụng được winsock ta cần phải cài đặt winsock DLL bằng việc gọi hàm WSAStartup. Sau khi làm việc xong với winsock ta cần giải phóng nó bằng cách gọi hàm WSACleanup.

Để kết nối tới server thông qua winsock ta cần phải chuyển đổi hostname của server thành cấu trúc addrinfo bằng cách sử dụng hàm getaddrinfo. Hàm này sẽ trả về một con trỏ tới danh sách các addrinfo, ta cần kết nối lần lượt từng địa chỉ trong danh sách này cho đến khi kết nối thành công.

Ta sử dụng hàm socket để khởi tạo một socket và sử dụng nó cho việc kết nối tới server. Sau khi một socket được tạo thành công ta gọi hàm connect để yêu cầu kết nối tới server thông qua socket này.



Hình 17: kết nối đến server thông qua winsock

Sau khi kết nối đến server thông qua socket, ta sử dụng các hàm send và recv để gửi và nhận dữ liệu với server, khi kết thúc quá trình làm việc với server ta cần đóng socket bằng việc sử dụng hàm closesocket.

### Lấy địa chỉ MAC của máy người dùng bằng giao thức ARP

Chương trình sử dụng giao thức ARP để xác định địa chỉ MAC của máy victim. Để thực hiện được điều này chương trình sử dụng hàm API có sẵn đó là SendARP, hàm này được cung cấp trong bộ API winsock vì thế ta cần phải khởi tạo winsock DLL trước khi dùng nó bằng cách gọi hàm WSAStartup.

# PHẦN III: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## CHƯƠNG 1: TRIỂN KHAI

Đề tài được chia làm hai phần là client(spy) và server, mỗi phần có cách triển khai khác nhau và cụ thể như sau:

### Triển khai client(spy)

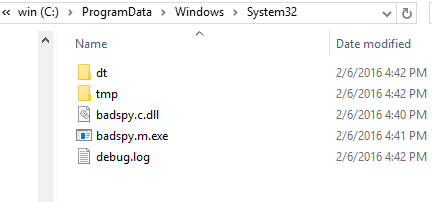
* 1. Yêu cầu hệ thống

Chương trình spy không yêu cầu cài đặt thêm bất cứ thư viện hổ trợ nào khác. Các phiên bản hệ điều hành Windows được hổ trợ bao gồm Windows XP, Windows Vista, Windows 7, Windows 8/8.1 và Windows 10.

* 1. Cài đặt

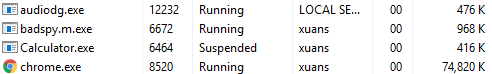
Spy tự động cài đặt chính nó và khởi chạy khi người dùng click double vào file thực thi của spy. Theo mặt định nó sẽ tự động cài đặt vào thư mục ẩn của hệ thống theo đường dẩn: C:\ProgramData\Windows\System32. Chú ý rằng đường dẩn có thể khác tùy thuộc vào cấu hình biến môi trường của Windows. Sau khi spy chạy lần đầu nó sẽ tự động đăng ký với registry để Windows khởi chạy nó mỗi khi vào hệ điều hành. Vì spy chạy ngầm trong hệ thống nên người dùng không thể biết sự hiện diện của nó trừ khi sử dụng các chương trình chuyên biệt như Task Manager, Process Hacker…

Hình ảnh bên dưới là ảnh chụp thư mục ẩn mà spy được cài đặt, như đã thấy thì nó sẽ tự động tạo ra 2 folder để lưu tạm các dữ liệu log và dữ liệu các gói.

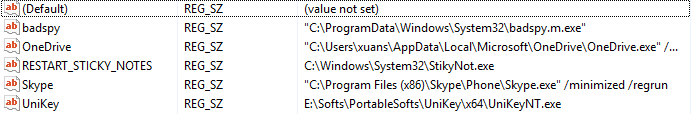


Hình 18: thư mục mà spy tự động cài vào

Trong thư mục này có một file debug.log chứa toàn bộ log của spy nhằm mục đích debug. Khi ở bản release thì toàn bộ code ghi log trong spy sẽ bị xóa lúc tiền biên dịch nhằm tăng tốc độ hoạt động của spy.



Hình 19: sử dụng task manager để xác định spy có đang chạy không



Hình 20: sử dụng tiện ích regedit của Windows để xác định spy có đang tự khởi động không

### Triển khai server

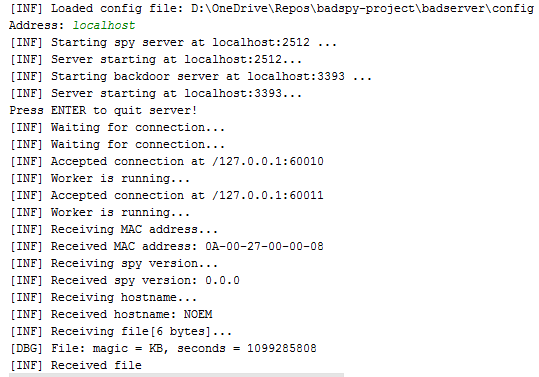
* 1. Yêu cầu hệ thống

Server được viết trên ngôn ngữ java vì thế yêu cầu máy chạy server phải được cài đặt sẵn java runtime. Cũng vì thế mà server có thể chạy trên nhiều hệ điều hành khác nhau như Windows, Linux…

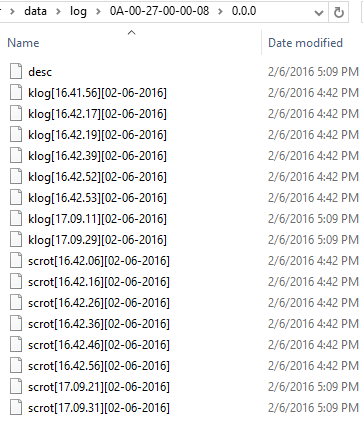
* 1. Cài đặt

Server chỉ bao gồm một file thực thi có đuôi jar kèm theo một file text cấu hình(file này sẽ được server tạo ra nếu chưa tồn tại). Đa số thông tin cấu hình sẽ được nạp từ file này trừ thông tin về địa chỉ IP của server sẽ lắng nghe trên đó. Vì thế ta cần nhập địa chỉ này để server có thể hoạt động. Có thể chạy server bằng cách sử dụng dòng lệnh: java –jar badserver.jar

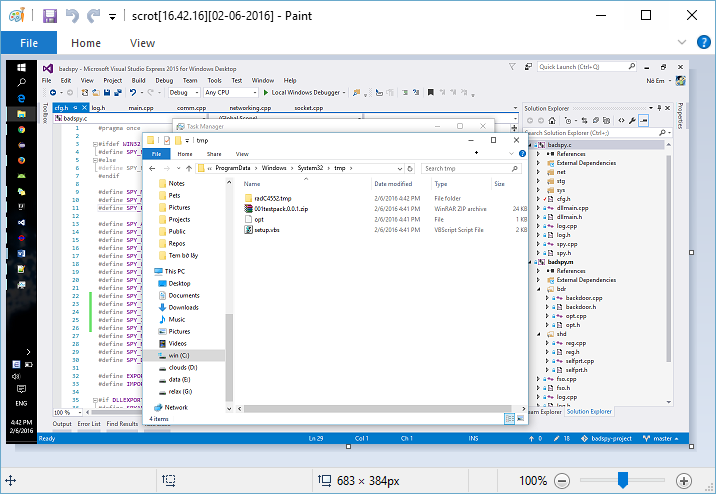
Sau khi chạy tại dấu nhắc lệnh ta cần nhập IP của server, để tắt server ta chỉ cần nhấn enter.



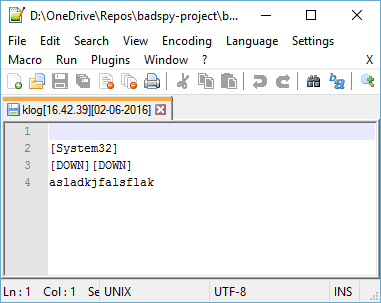
Hình 21: log được ghi lại tại server nhằm mục đích debug



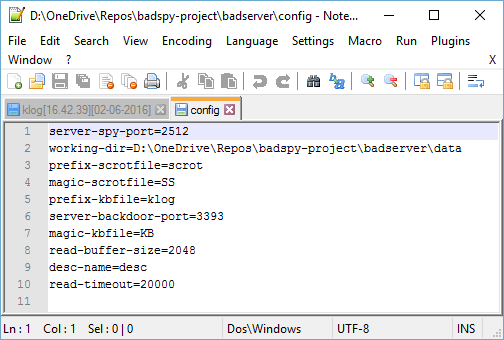
Hình 22: log file được ghi lại



Hình 23: một screenshot file ghi lại ảnh chụp màng hình của người dùng



Hình 24: đây là log file ghi lại phím bấm của người dùng



Hình 25: file config của server

## CHƯƠNG 2: ĐÁNH GIÁ KẾT QUẢ

Spy đã hoạt động đúng ý đồ, không những thu thập dữ liệu hiệu quả mà còn mở backdoor cho các chương trình độc hại khác xâm nhập về sau. Tốc độ upload và download còn chưa cao do log ghi ra quá nhiều để phục vụ quá trình debug.

.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Spy có kích thước nhỏ, tốc độ thực thi nhanh và chiếm ít bộ nhớ vì thế nó dể dàng được cài đặt vào máy victim. Log được spy ghi lại được lưu trữ cục bộ nếu nó không kết nối tới server được, các log này sẽ được tự động đẩy lên server một khi spy kết nối lại được với server. Spy ghi log toàn bộ các phím được nhấn và chuyển đổi nó thành các tên dể hiểu kèm theo tên cửa sổ mà phím được nhấn tương ứng, như thế việc phân tích dữ liệu của victim về sau trở nên đơn giản hơn. Kích thước ảnh chụp được scale nhỏ lại nhằm giảm dung lượng lưu trữ và tăng tốc độ upload lên server.

Hiện tại spy chưa có cơ chế bảo vệ process cũng như ẩn process khỏi Task Manager vì thế nên nó dể dàng bị tắt bởi người dùng có hiểu biết. Ảnh chụp màng hình được lưu dưới dạng bitmap không nén vì thế kích thước còn lớn.

Một số hướng phát triển của spy như sau:

* Nén và mã hóa dữ liệu log để tăng tốc độ upload và bảo mật cũng như giảm dung lượng lưu trữ.
* Thay vì phát triển spy là một ứng dụng Windows bình thường thì ta có thể xây dựng theo hướng là một Windows service để tăng khả năng ẩn nấp của spy. Đồng thời sử dụng cơ chế song song hai tiến trình khác nhau để bảo vệ nhau, khi một tiến trình bị ngắt thì tiền trình kia phát hiện và khởi chạy lại tiến trình bị ngắt đó. Với cơ chế này thì victim khó khăn hơn trong việc xử lý spy.
* Tự động thu thập các file tài liệu trong máy victim đồng thời cung cấp khả năng âm thầm ghi lại webcam và microphone trên máy victim.
* Ngăn chặn máy victim cài đặt các chương trình antivirus cũng như tự động thêm spy vào danh sách ngoại lệ của tường lửa.

# KẾT LUẬN CHUNG

Từ việc nguyên cứu, xây dựng một spy hoàn chỉnh ta có thể hiểu được cơ chế hoạt động cũng như cách phát hiện một chương trình độc hại. Từ đó có những chính sách bảo mật cho hệ thống thích hợp để tránh nhiễm phải các chương trình độc hại như spy chẳng hạn. Và ta cũng cần biết rằng không có một hệ thống nào là bảo mật tuyệt đối và đôi khi chính chúng ta lại gián tiếp/trực tiếp cài đặt những thứ nguy hiểm vào hệ thống mà ta không hề hay biết, hoặc chính hệ thống mà ta đang xài lại là mối nguy hiểm nhất. Cũng chính vì thế mà việc xây dựng được một hệ thống bảo mật tuyệt đối là điều không thể, ngay cả hệ điều hành ta đang xài, trình duyệt, các ứng dụng, mạng xã hội…tất cả đang âm thầm thu thập thông tin cá nhân mà ta không hề hay biết vì trên thế giới này chằng có gì là miễn phí cả.

Các kiến thức về hệ điều hành cũng như về mạng có thể xây dựng được những ứng dụng mang tính thực tế cao, phục vụ cho nhiều lỉnh vực đời sống từ đó nâng cao chất lượng cuộc sống con người. Nhưng tồn tại không ít những mặt tối, một số người lợi dụng những hiểu biết này để xây dựng những chương trình độc hại nhằm lấy cắp thông tin người dùng, một số khác thích thể hiện bằng cách phá hoại máy người dùng các chương trình virus làm hư hỏng hệ thống, các chương trình spam…. Do đó người dùng phải luôn cẩn trọng khi sử dụng máy tính và mạng internet như không click vào các link lạ, không cài đặt phần mềm từ các nguồn không tin cậy,…..

# TÀI LIỆU THAM KHẢO

https://en.wikipedia.org/wiki/Windows\_Registry

https://en.wikipedia.org/wiki/Thread\_(computing)

https://en.wikipedia.org/wiki/Batch\_file

https://vi.wikipedia.org/wiki/JDK

https://en.wikipedia.org/wiki/Transmission\_Control\_Protocol

https://en.wikipedia.org/wiki/Network\_socket

https://vi.wikipedia.org/wiki/Client-server

https://en.wikipedia.org/wiki/Winsock

https://en.wikipedia.org/wiki/Address\_Resolution\_Protocol

# PHỤ LỤC