Lab 12-03

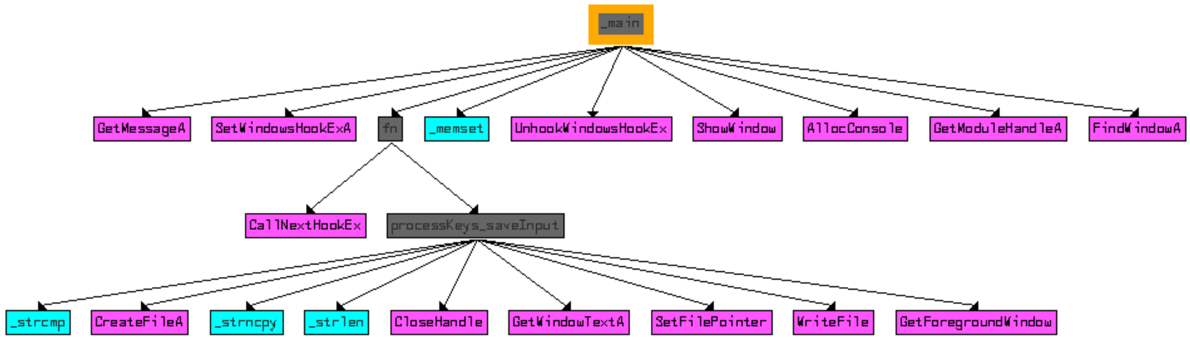
Analyze the malware extracted during the analysis of Lab 12-2, or use the file Lab12-03.exe.

Questions

1. What is the purpose of this malicious payload?

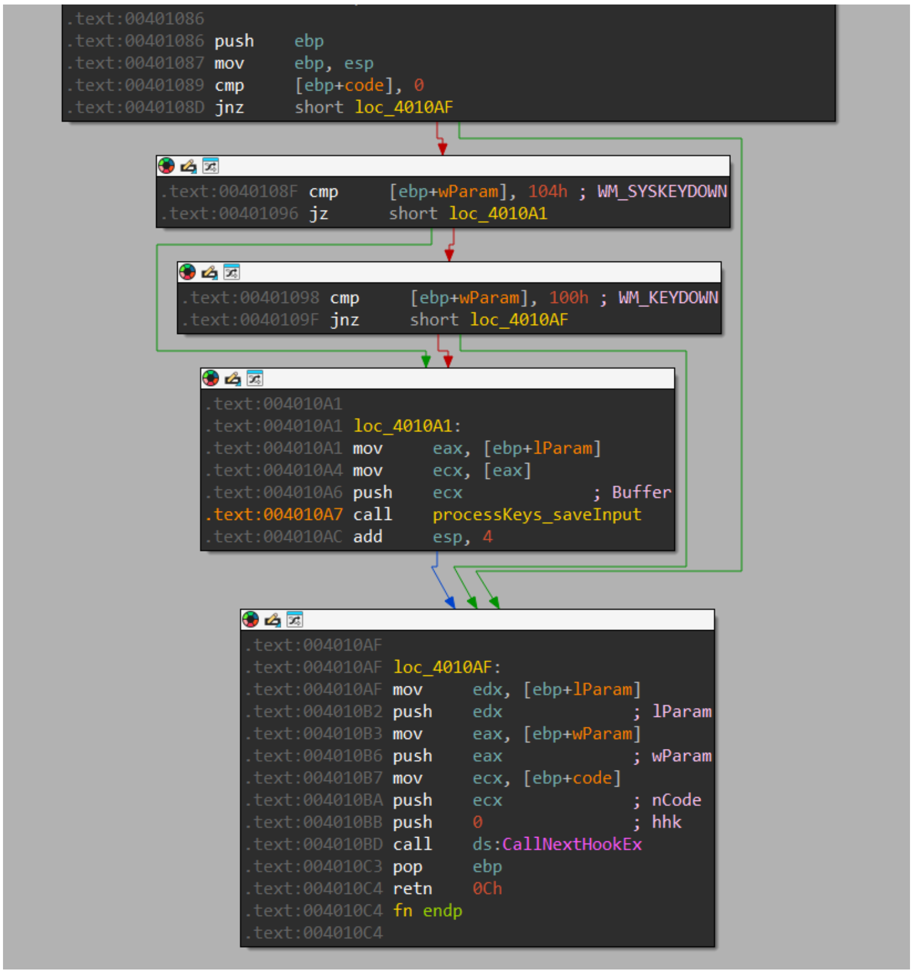
The purpose of this malware is to capture keyboard input with a corresponding window title before the application gets it. This way it can manipulate the input without the user’s knowledge.

2. How does the malicious payload inject itself?



When it comes to its behavior, it captures keystrokes through hooking using **SetWindowsHookExA** with an ID **0x0D** that stands for **WH_KEYBOARD_LL.**

SetWindowsHookExA performs a hook procedure that IDA named **fn**, that records all of the keys and its correlated window names using **GetForegroundWindow**, **GetWindowTextA** and after all it saves the output continuously to a file **practicalmalwareanalysis.log**.



```
.text:00401086
.text:00401086 push    ebp
.text:00401087 mov     ebp, esp
.text:00401089 cmp     [ebp+code], 0
.text:0040108D jnz     short loc_4010AF
```

```
.text:0040108F cmp     [ebp+wParam], 104h ; WM_SYSKEYDOWN
.text:00401096 jz      short loc_4010A1
```

```
.text:00401098 cmp     [ebp+wParam], 100h ; WM_KEYDOWN
.text:0040109F jnz     short loc_4010AF
```

```
.text:004010A1
.text:004010A1 loc_4010A1:
.text:004010A1 mov     eax, [ebp+lParam]
.text:004010A4 mov     ecx, [eax]
.text:004010A6 push    ecx             ; Buffer
.text:004010A7 call    processKeys_saveInput
.text:004010AC add     esp, 4
```

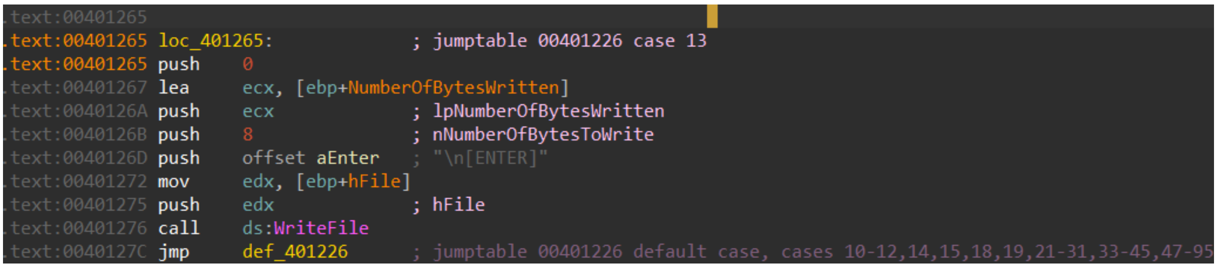
```
.text:004010AF
.text:004010AF loc_4010AF:
.text:004010AF mov     edx, [ebp+lParam]
.text:004010B2 push    edx             ; lParam
.text:004010B3 mov     eax, [ebp+wParam]
.text:004010B6 push    eax             ; wParam
.text:004010B7 mov     ecx, [ebp+code]
.text:004010BA push    ecx             ; nCode
.text:004010BB push    0               ; hhk
.text:004010BD call    ds:CallNextHookEx
.text:004010C3 pop     ebp
.text:004010C4 retn    0Ch
.text:004010C4 fn endp
.text:004010C4
```

In more details, the malware compares the input key with 0x104 and 0x100 which corresponds to **WM_SYSKEYDOWN** and **WM_KEYDOWN**, that way it knows when to process all of the pressed keys at once.

If the pressed key is the same as one of these two, that means the input is done and should be processed, the processing takes place at what I called **processKeys_saveInput** at **0x4010A7**.

Here is the "core" of the keylogger that is processing the window name and the input.

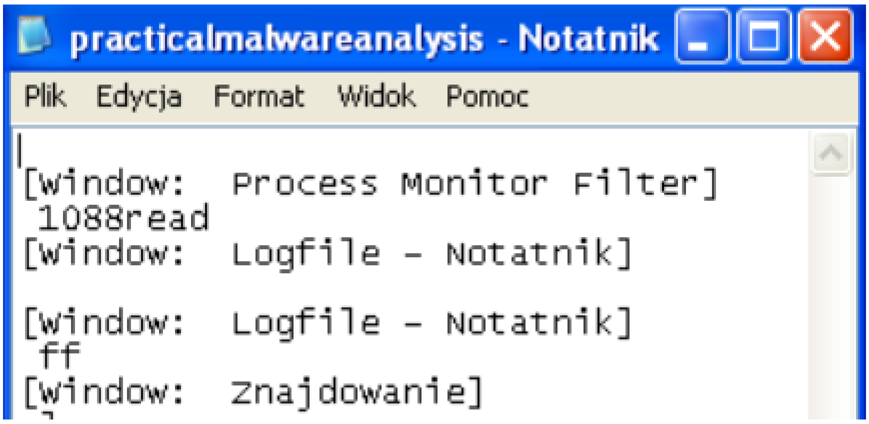
The input key is made by a switch jumptable with a size of 98 cases, here is an example of one that is detecting if the ENTER key is pressed, if it does, then it writes the output into the file **"[ENTER]"**.



```
.text:00401265
.text:00401265 loc_401265:                ; jumptable 00401226 case 13
.text:00401265 push    0
.text:00401267 lea     ecx, [ebp+NumberOfBytesWritten]
.text:0040126A push    ecx                ; lpNumberOfBytesWritten
.text:0040126B push    8                  ; nNumberOfBytesToWrite
.text:0040126D push    offset aEnter      ; "\n[ENTER]"
.text:00401272 mov     edx, [ebp+hFile]
.text:00401275 push    edx                ; hFile
.text:00401276 call    ds:WriteFile
.text:0040127C jmp     def_401226         ; jumptable 00401226 default case, cases 10-12,14,15,18,19,21-31,33-45,47-95
```

3. What filesystem residue does this program create?

The malware stores all of the stolen data in a newly created file **practicalmalwareanalysis.log** that is located at **Lab12-02.exe** current execution directory.



```
[Window:  Process Monitor Filter]
 1088read
[Window:  Logfile - Notatnik]

[Window:  Logfile - Notatnik]
 ff
[Window:  Znajdowanie]
```