Lab 12-02

Analyze the malware found in the file Lab12-02.exe.

File Analysis

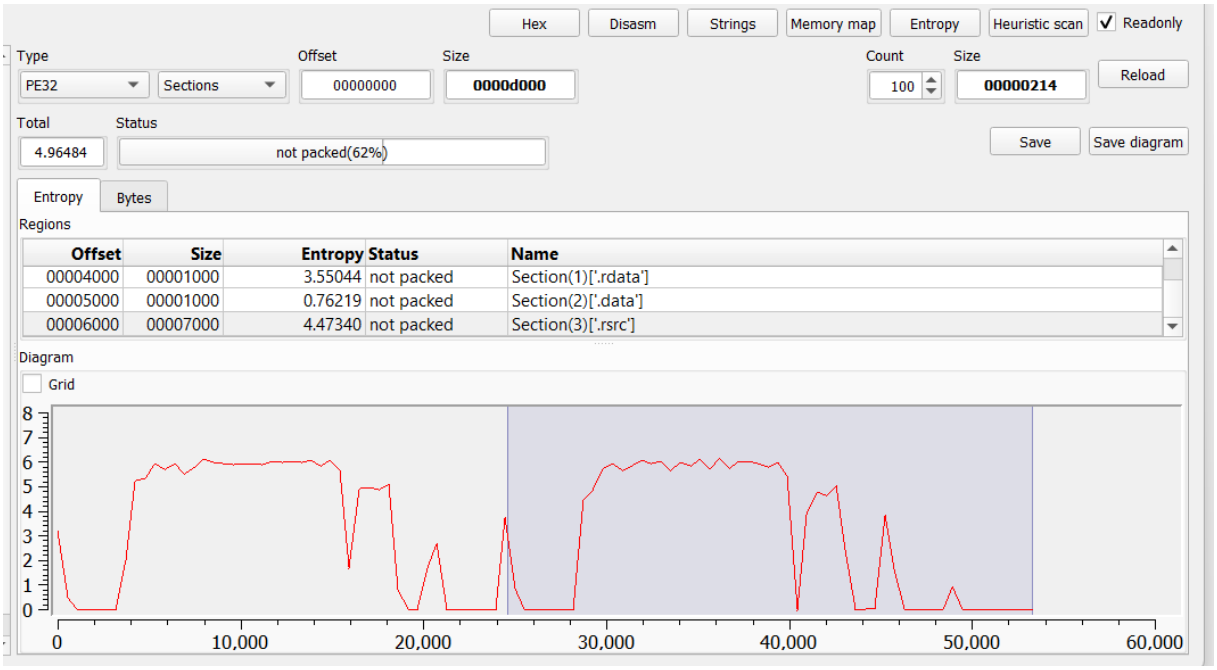
First of all, let's check what we are dealing with before execution.

Lab12-02.exe is a malware written using C++.

The file does not seem packed, entropy is not that high (almost 5), section names remain unchanged, virtual and physical sizes are almost the same.

When it comes to the topic of entropy, there are some suspicious peaks at resources section (highlighted area at image below), the 3 big peaks might tell us that there might be another file hiding there.

To confirm that, we have to take a look at it in further analysis.



Interesting flossed strings:

- \svchost.exe
- LOCALIZATION
- UNICODE

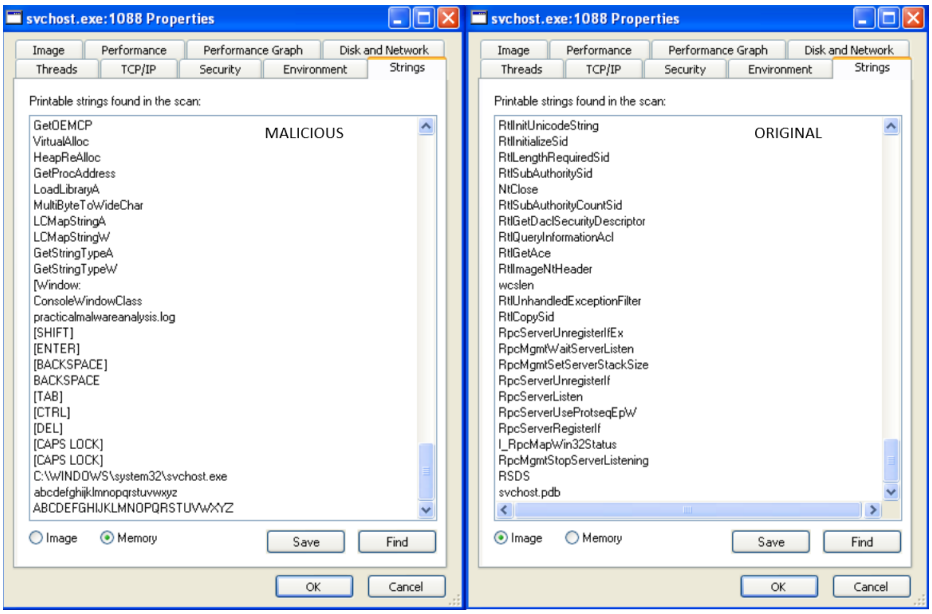
The malware declares only one library import (kernel32.dll), the functions hints us possible behavior of the sample:

- File operations
- Allocating memory, writing memory
- Thread management
- Process management
- Dynamically library loading
- Resources management

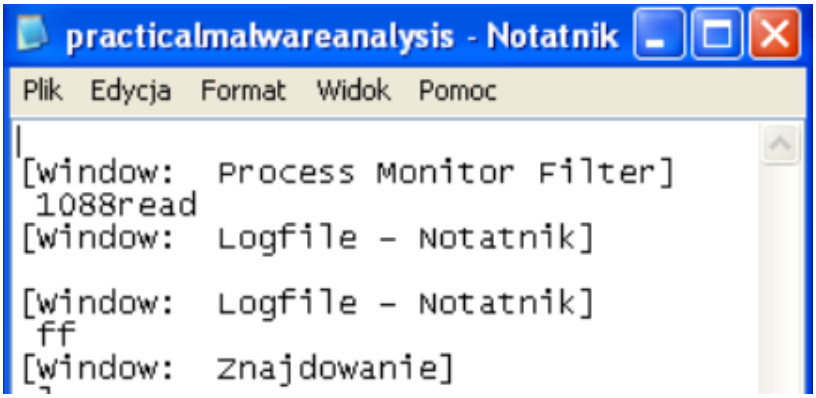
There is a section containing resources, inside it we have an entry “UNICODE/LOCALIZATION containing 24576 bytes of data. The data seems pretty random and does not reveal much at this point, for example there are three bigger loads of data (3504, 3263, 1928 bytes each) containing mostly 0x41, 0x40 values (corresponding A,B ASCII)

We will have to check at later steps how it resolves data after API calls to the resources.

In the svchost.exe memory strings there are some indicators that we are dealing with activity related to keylogging, there are bunch of keys such as [SHIFT], [ENTER], there is also a string path to the victim process and we have a possible name of a file storing gathered data **practicalmalwareanalysis.log**.



The .log file is being saved in the **Lab12-01.exe** directory, it saves the data about currently running window and it's keystrokes.

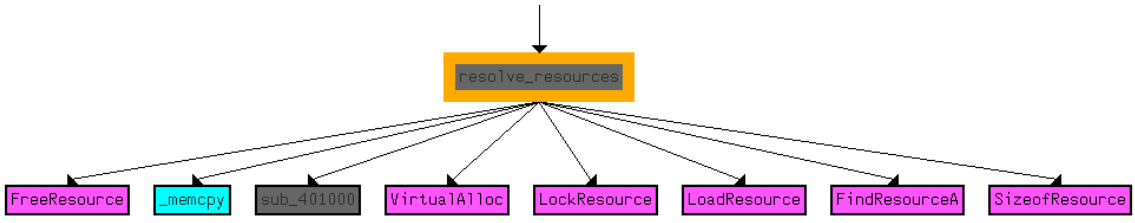


In the threads, there is no sign of any DLLs attached and the process was newly created after file execution, so it might process hollowing attack using suspension state.

Let's when we have some knowledge about it, let's put it now into disassembler to see what is does exactly.

The entry point for the executable is located at 0x4014E0, the code resolves system directory path using **GetSystemDirectoryA**, connects the string with \\svchost.exe to form a path to a victim file.

Later, at 0x401528 we have a call to a sub that I named **resolve_resources**, as the name says, this function locates **UNICODE/LOCALIZATION** resource and allocates it in the memory of the currently running process for later usage.

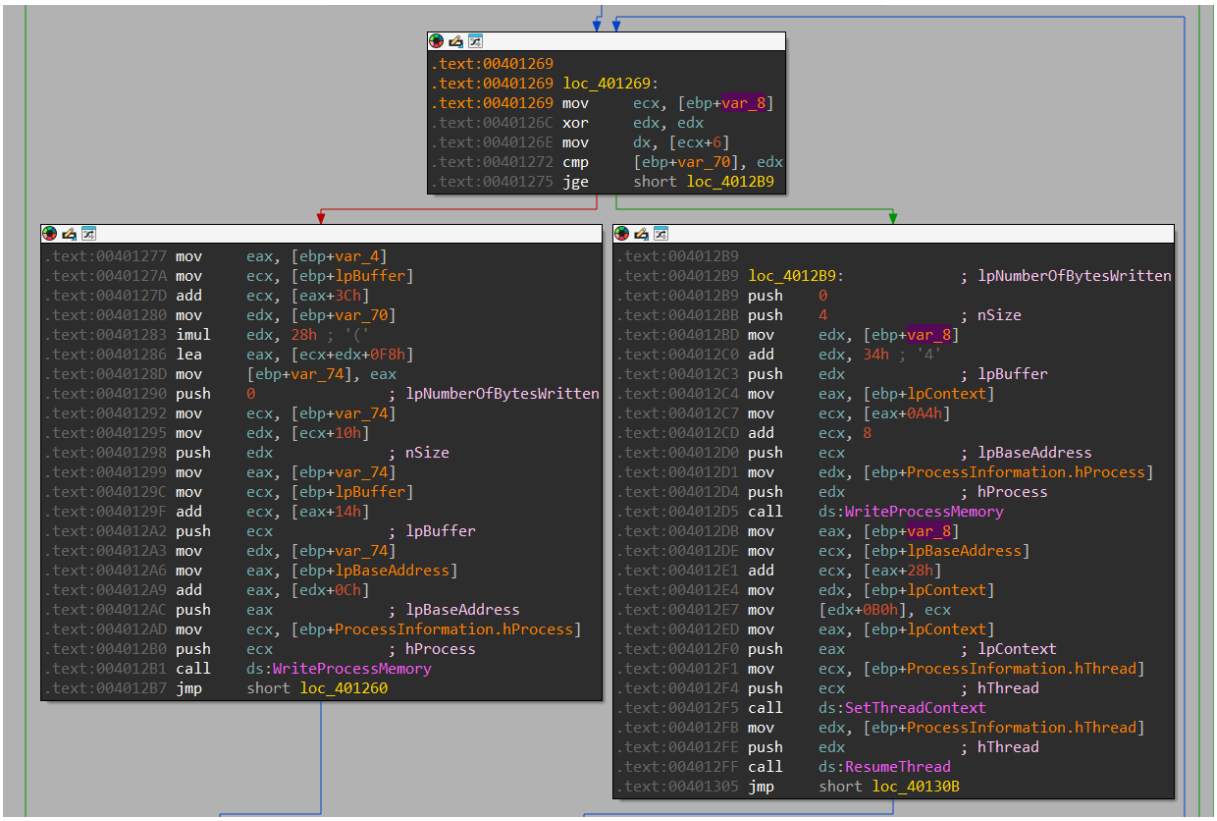


After we gathered path to the victim, resolved the resources it calls another function that I called **start_svchostWithResource**.

The function mainly focuses at implementing process hollowing. The attack divides to few steps: starting the victim process in a suspended state (dwCreationFlags set to 4), unmapping current victim memory using dynamically imported ntdll.dll function **NtUnmapViewOfSection**, allocating space in the victim memory (**VirtualAllocEx**), filling the space with the new code (**WriteProcessMemory**).

The writing call is used PE header and for each of the sections.

All that's left is to set the register by **SetThreadContext**, and execute the code with **ResumeThread** call.



In order to fully analyze the malware, we have to find the data that gets resolved and hollowed into svchost.exe.

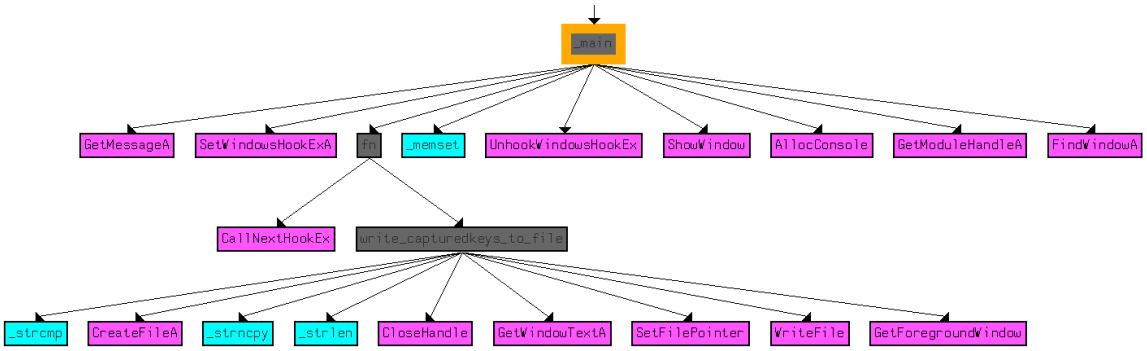
To get it, we have to locate the data after its being resolved and decoded and then dump it.

Dump PE file

The dumped PE file seems not protected by any additional layer of obfuscation or packing.

When it comes to its behavior, it captures keystrokes through hooking using SetWindowsHookExA with an id 0x0D that stands for WH_KEYBOARD_LL which confirms our previous dynamic analysis when we saw each pressed key being saved into the file.

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



Questions

1. What is the purpose of this program?

The purpose of this malware is to record all of the keystrokes and its corresponding window names. It's a loader and keylogger type of malware.

2. How does the launcher program hide execution?

The launcher hides execution by performing process hollowing.

It creates a suspended svchost.exe process, unmaps existing memory and overwrites it with a completely different code of its own PE file and resumes the process.

3. Where is the malicious payload stored?

The malicious payload is hidden in resources section under **UNICODE/LOCALIZATION** of the provided in labs PE file.

At the stage of execution it gets resolved from the section and then mapped into the victim process.

4. How is the malicious payload protected?

The malicious encoded payload is first resolved from the resources at 0x40132C and then randomly allocated in the memory.

The decoding function takes place at 0x401000, each of the byte is XOR-encoded by 0x41, so the code just performs XOR operation on each byte of the resolved resource with 0x41.

5. How are strings protected?

As explained above – the strings are XOR-encoded with value of 0x41, to extract them we have to perform the same operation.