Malware reconciliation design (Part 4)

In the previous sections of this article, I have shown you how to receive and open a packaged UPX file. In the next section, we will actually look at a malware sample in the uncompressed format of the software

Malware reconciliation design (part 1)
Malware counterpart design (part 2)
Malware counterpart design (part 2)

In the previous sections of this article, I have shown you how to receive and open a packaged UPX file. In this next article we will actually look at a malware sample in its uncompressed format.

In the last part of this collation design we will go into a look at an open malware pattern. There are a number of different purposes for implementing contrast designs and there are also a number of different methods. However, in our case where malware analysis is being done, what we want to draw from this analysis is a deeper, deeper understanding of what malware does and some other areas. We still use both dynamic and static principles as described in the previous sections.

Implementing reverse engineering to deploy an example is absolutely no harm. For each example you perform an analysis on an existing program such as Windows XP or some other FTP server program. What you look for is also different. You might be looking for any examples that might cause buffer overflows, string formatting issues, and code vulnerabilities related to it. To do this you really have to go to each of the first operands in the subroutine. This problem may sound difficult and time-consuming and in addition it also requires a good understanding of programming. For simplicity with such malware, you can use dynamic and static methods of collation design to solve.

As you can see, there are a number of different reasons why you want to implement reverse engineering and many other purposes. Tools for this work include debugger debugger, disassembler, and hex editor. With these tools, we can fully embark on the analysis of a packaged malware. They will have to perform a lot of procedures, but we will present to you relatively easy to understand information.

Find malware

You can see the section we mentioned earlier on installing the Malcode Analyst Pack from Idefense. What we do first in analysis is to run a command 'strings' from the MAP tool for malware that has been packaged. All you need to do is right-click on the malware and the 'strings' option will be displayed. Click to select this option. When the command is executed, another window will appear as shown below.

Figure 1

We can see in the above window that there are some first items that go along with MZ header and some other parts such as .text, .data and .idata. Also listed here is a bunch of MD5 file miscellaneous and its size. Now drag the scroll bar to see the output sequence we care about appearing in binary.

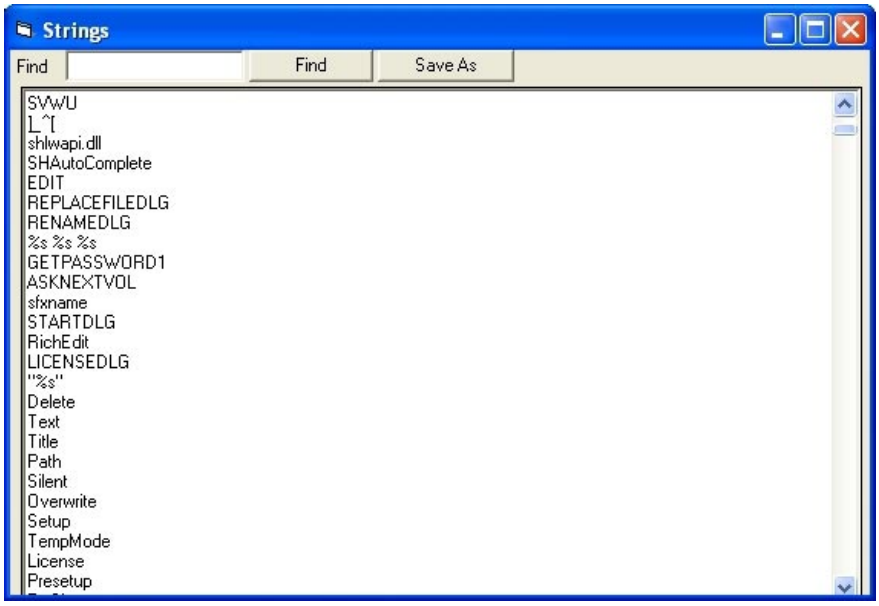

Figure 2

One of the items here that I find important is 'shlwapi.dll'. This dll file is really confusing so I go to Microsoft Technet to search for it and determine if it is malware. With a long list of vulnerabilities related to this dll and this is definitely a malware.

Figure 3

After reviewing it. We continue to scroll down to find binary malware. Most of them are written, opened, and nested with registration keys. There is also a long list of ASCII strings that appear to be malware samples, when executed will appear users with some type of window. I have made this assumption for the string 'CreateWindowExA' as shown below.

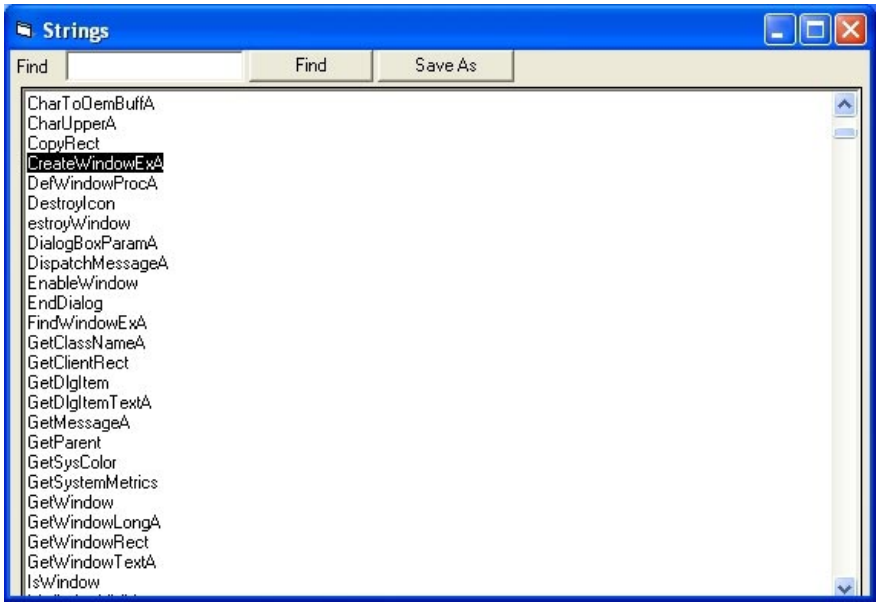

Figure 4

The spelling of those ASCII strings doesn't seem to be very troublesome, but try Google to see if we can give a better explanation of what it is. That's what I did as shown below. My suspicion has been confirmed, a window

will appear to the user when the malware is executed.

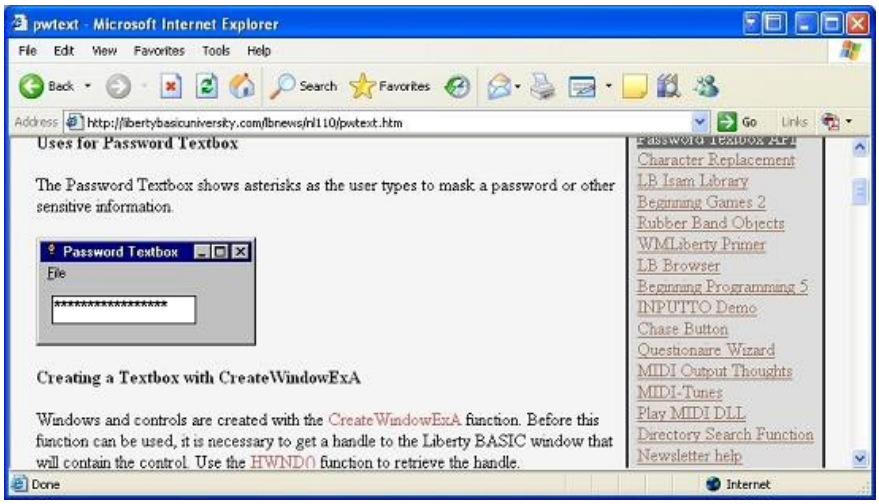

Figure 5

The remaining strings include a number of other words that are also needed to quickly find out if they are malware and what malware is. These actions are performed in a static phase. As you can see, there is a large amount of information that can be accumulated from binary malware.

From static to dynamic

I really want to find quite interesting things and go to the core of the problem. That made me move on to further studying the dynamic part of the analysis. You will have to install and run both regmon and filemon. Once installed and running, make sure you exclude all running items found in the two applications. That will help you catch all the actions that occur when malware comes into play. Here I do a simple job of renaming the malware with the extension .exe. I called them through a DOS command window. When you do that, a window will appear. Are you really interested in the strings it can draw from binary?

Figure 6

When we see the ASCII string CreateWindowExA and then use Google, we get our purpose. This is the code used to create the pop-up window that we see above. Now what to do with what happened in regmon and filemon after I called binary malware? Look at the two windows below.

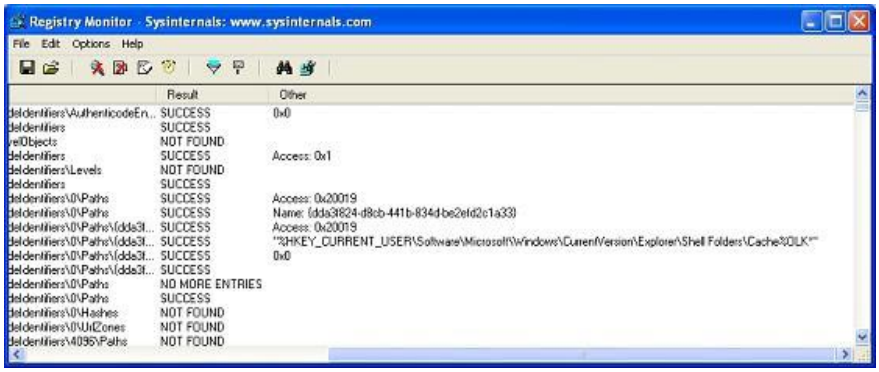

Figure 7

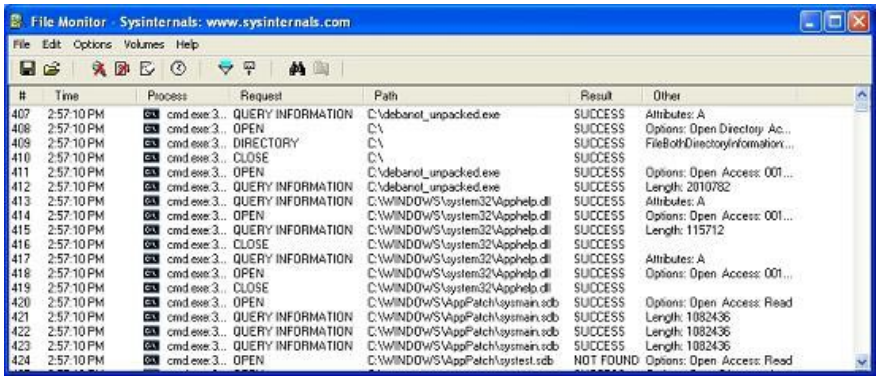

Figure 8

As we imagine, malware creates a cluster of changes to registry keys, writes some files and some other components that we associate with it. It is time-consuming to correctly analyze what is given in regmon and filemon. This will be done by checking that anything is inserted into the hard drive. This action is quite old with malware, usually they still use trojans to maintain computer access. Or you have downloaded Spyware, or another type of malware from the website or ftp server. Nothing here is really new, because it has been well documented on computer security pages.

Conclude

In this section you have seen the implementation of reverse engineering for many purposes without the need for too much complicated effort. All that needs to be done is to review and access some pages on the Internet that have malware and how to analyze it. Through this article, we hope it will be helpful to you and look forward to receiving your feedback on this issue.