LABS

**Lab 6-1**

In this lab, you will analyze the malware found in the file *Lab06-01.exe*.

Questions

1. What is the major code construct sound in the only subroutine called by ***main***?
2. What is the subroutine located at 0x140001066 and 0x140001050?
3. What is the purpose of this program?

**Lab 6-2**

Analyze the malware found in the file ***Lab06-02.exe***.

Questions

1. What operation does the first subroutine called by main perform?
2. What is the subroutine located at 0x14000103B?
3. What does the second subroutine called by main do?
4. What type of code construct is used in this subroutine?
5. Are there any network-based indicators for this program?
6. What is the purpose of this malware?

**Lab 6-3**

In this lab, we’ll analyze the malware found in the file ***Lab06-03.exe*.**

Questions

1. Compare the calls in main to Lab 6-2’s main method. What is the new function called from main?
2. What parameters does this new function take?
3. What major code construct does this function contain?
4. What can this function do?
5. Are there any host-based indicators for this malware?
6. What is the purpose of malware?

**Lab 6-4**

In this lab, we’ll analyze the malware found in the file *Lab06-04.exe*.

*Questions*

1. What is the difference between the calls made from the main method in  
   Labs 6-3 and 6-4?
2. What new code construct has been added to main?
3. What is the difference between this lab’s parse HTML function and  
   those of the previous labs?
4. How long will this program run? (Assume that it is connected to the  
   Internet.)
5. Are there any new network-based indicators for this malware?
6. What is the purpose of this malware?

**Lab 6-1 Solutions**

**Short Answers**

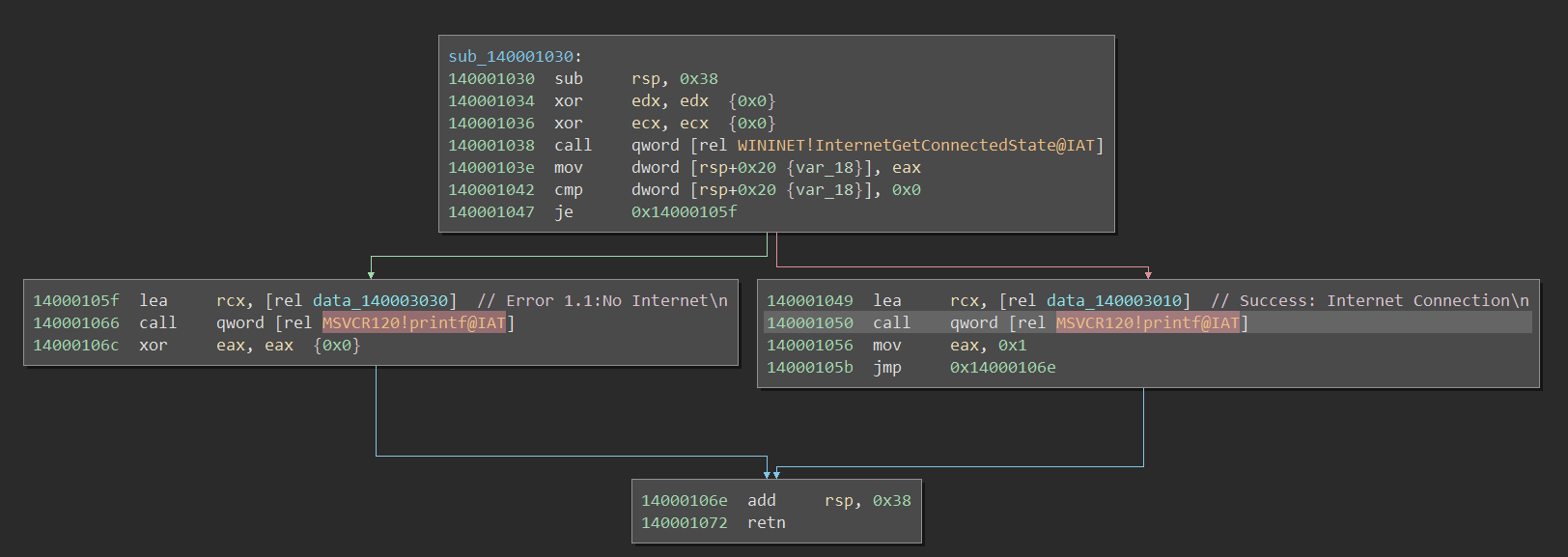
1. The majority of the code is an if statement located at address 0x140001030.
2. The *printf* is located at 0x140002020.
3. The program checks for an active internet connection. if active internet connection is found then it prints “Success: Internet Connection.”. if a connection is not found, it prints “Error 1.1: No Internet.”

Detail Analysis

We began the analysis by dragging the file in to binary ninja and see the assembly. We see the WININET call which tells us that we are using Windows Internet (WinINet) API in this exercise.

Using MSDN, we learn this Windows API function checks the status of the internet connection for the local system. The Strings “Success: Internet Connection.” Or “Error 1.1: No Internet.”

The main function for the file by clicking the \_start in the left side panel. The main function calls a subroutine located at the 0x140001030. The clicking the subroutine we can go to the subroutine. The following picture shows the subroutine disassembly.



The sub\_140001030 is where the program used the InternetGetConnection API to check the internet connection state, before the je instruction it compares the value returned from the InternetGetConnection. If the value returned is 1 than it will print Success: Internet Connection or if the value is 0 it will print Error 1.1: No Internet.

**Lab 6-2 Solutions**

**Short Answers**

1. The first subroutine is at a 0x140001060, the subroutine checks the internet status on the local machine.
2. Printf is located at 0x14000103B.
3. The second subroutine is located at the 0x1400010b0. It downloads the web page located at: <http://www.practicalmalwareanalysis.com/cc.htm>and parses an HTML comments from the beginning of the page.
4. This subroutine uses a character array filled with data from the call to InternetReadFile. This array is compared one byte at a time to parse an HTML comment.
5. There are two network-based indicators. The program uses the HTTP User-Agent Internet Explorer 7.5/pma and downloads the web page located at: <http://www.practicalmalwareanalysis.com/cc.htm>.
6. The program checks for the active Internet connection. If none is found, the program terminates. Otherwise the program attempts to download a web page. The webpage contains an embedded HTML comment starting with <!--. The next character is parsed from this comment and printed on the screen in the format “Success: Parsed command is X” where X is the character parsed from the HTML comment. If successful, the program will sleep for 1 minute and then terminates.

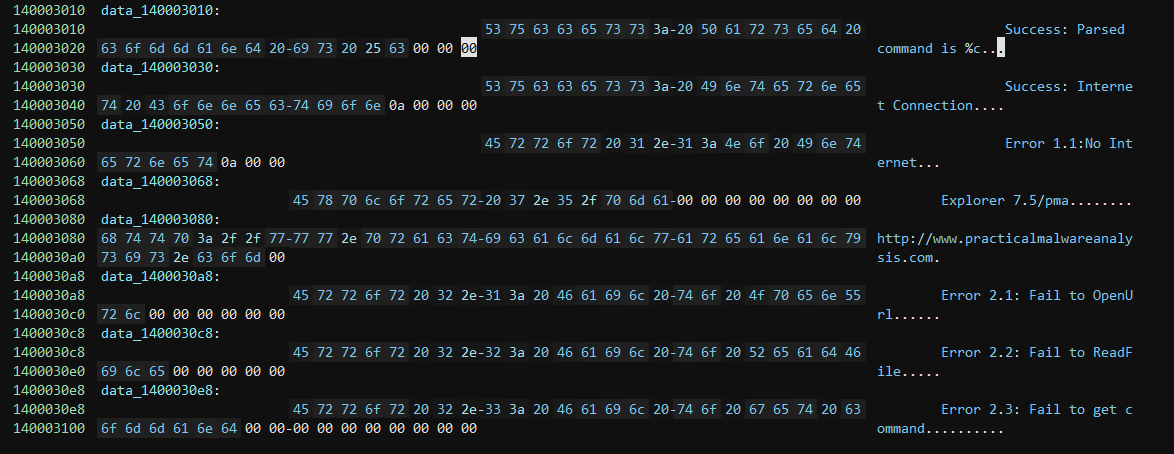
Detailed Analysis

We start by loading the file in Binary ninja and navigating the left side panel, we can see the file has more subroutine then the previous exercise. We can press space to see the hex disassembly of the file do see the full program. This can help us scan for obvious string and data that can give us a hint of what the program is doing.

Navigating to address 0x14000200 we can see the list of the Windows internet API being used in this program, the program is using following API calls.

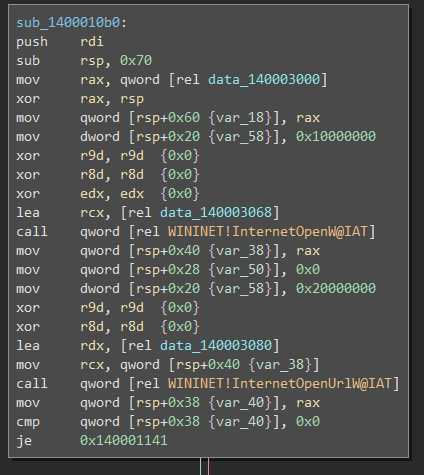
|  |
| --- |
| ***InternetGetConnectedState***  ***InternetOpen***  ***InternetReadFile***  ***InternetOpenUrl*** |

Scroll down more and that revels the data that are stored in the memory this includes the sting that are being used in the program the following picture shows the strings that are stored in the memory. We notice the Error messages and the URL *http://www.practicalmalwareanalysis.com/cc.htm*. that the program tries to make the HTTP request.

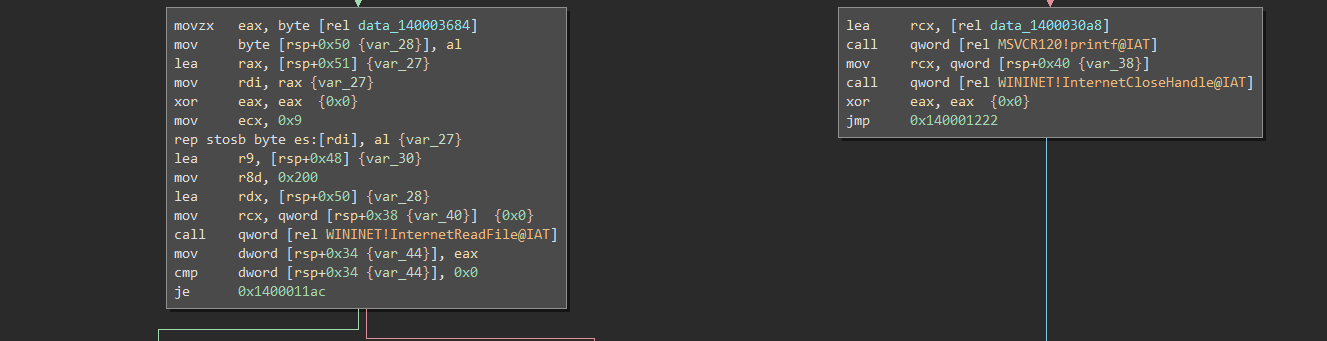


All of these functions are part of WinINet, a simple API for using HTTP over a network. They work as follows:

* InternetOpenA is used to initialize the use of the WinINet library, and it sets the User-Agent used for HTTP communication.

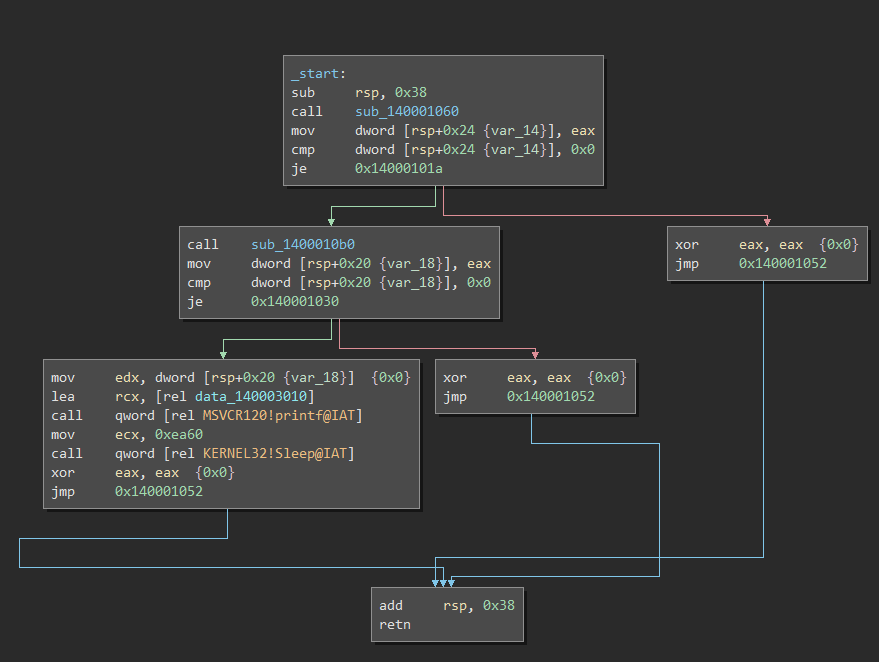


* InternetOpenUrlA is used to open a handle to a location specified by a complete FTP or HTTP URL. (Programs use handles to access something that has been opened. We discuss handles in Chapter 7.)
* InternetReadFile is used to read data from the handle opened by InternetOpenUrlA.



* InternetCloseHandle is used to close the handles opened by these files.

To summarize, this program checks for an active Internet connection, and then downloads a web page containing the string <!--, the start of a comment in HTML. An HTML comment will not be displayed in a web browser, but you can view it by looking at the HTML page source. This technique of hiding commands in HTML comments is used frequently by attackers to send commands to malware while having the malware appear as if it were going to a normal web page.



**Lab 6-3 Solutions**

**Short Answers**

1. The functions at 0x140001080 and 0x1400010d0 are the same as those in Lab 6-2. At 0x140002068 is printf. The 0x140001260 function is new to this lab.
2. The new function takes two parameters. The first is the command character parsed from the HTML comment, and the second is the program name "Lab06-03.exe".
3. The new function contains a switch statement.
4. The new function can print error messages, delete a file, create a directory, set a registry value, copy a file, or sleep for 100 seconds.
5. The registry key Software\Microsoft\Windows\CurrentVersion\Run\Malware and the file location C:\Temp\cc.exe can both be host-based indicators.
6. The program first checks for an active Internet connection. If no Internet connection is found, the program terminates. Otherwise, the program will attempt to download a web page containing an embedded HTML comment beginning with <!--. The first character of the comment is parsed and used in a switch statement to determine which action to take on the local system, including whether to delete a file, create a directory, set a registry run key, copy a file, or sleep for 100 seconds

**Lab 6-4 Solutions**

**Short Answers**

1. The function at 0x1400010a0 is the check Internet connection method, 0x1400010f0 is the parse HTML method, 0x140002070 is printf, and 0x1400012e0 is the switch statement.
2. A for loop has been added to the main method.
3. The function at 0x1400010f0 now takes a parameter and calls sprintf with the format string Internet Explorer 7.50/pma%d. It builds a User-Agent for use during HTTP communication using the argument passed in.
4. This program will run for 1440 minutes (24 hours).
5. Yes, a new User-Agent is used. It takes the form Internet Explorer 7.50/ pma%d, where %d is the number of minutes the program has been running.
6. First, the program checks for an active Internet connection. If none is found, the program terminates. Otherwise, the program will use a unique User-Agent to attempt to download a web page containing a counter that tracks the number of minutes the program has been running. The web page downloaded contains an embedded HTML comment starting with <!--. The next character is parsed from this comment and used in a switch statement to determine the action to take on the local system. These are hard-coded actions, including deleting a file, creating a directory, setting a registry run key, copying a file, and sleeping for 100 seconds. This program will run for 24 hours before terminating.