Lab 11-02

Analyze the malware found in Lab11-02.dll. Assume that a suspicious file named Lab11-02.ini was also found with this malware.

Questions

1. What are the exports for this DLL malware?

In the file Lab11-02.dll there is only one export, called “installer”.

2. What happens after you attempt to install this malware using rundll32.exe?

After running the command “rundll32.exe Lab11-02.dll installer” the malware copies its original “Lab11-02.dll” file to system directory under a name “spoolwxx32.dll” and sets newly created filename to a value in AppInit_DLLs.

3. Where must Lab11-02.ini reside in order for the malware to install properly?

In order for the malware to install properly the file Lab11-02.ini must be located at system32 directory.

The DLL_MAIN part first checks for the file existence, then for the bytes size inside the file. If the bytes are higher than 0, then it proceeds to move further in the code.

4. How is this malware installed for persistence?

The trick in gaining persistence in this example is through exploiting the ability to attach any library located in system directory to a process that loads user32.dll.

In other words, whenever a process loads user32.dll (which is quite frequently) then the infected dll “spoolwxx32.dll” also gets loaded to that process.

5. What user-space rootkit technique does this malware employ?

The malware employs a technique called "inline hooking." This method modifies the start of a function in a legitimate library, redirecting it to malicious code. After executing its own operations, the malware runs the original code, restoring the function and redirecting control back to the legitimate code. As a result, everything appears normal from the user's perspective, and the system functions as expected.

6. What does the hooking code do?

As I said previously at answer 4, there are lots of processes that load user32.dll, in this example, the malware intention is to redirect outgoing emails to its decrypted e-mail address, so it must look for processes that are mail apps, it looks for:

- OUTLOOK.EXE
- THEBAT.EXE
- MSIMN.EXE

If the caller for the infected library is any of the ones mentioned above, then it prepares inline hook attack by changing first 5 bytes of the ws32.dll send function to JMP 0x1000113D. Then after the malicious code execution is done, it executes first "stolen" 5 bytes of the original send function, then redirects back to the send address and shifts 5 bytes forward.

The hooking code located at 0x1000113D scans every parameter for a string "RCPT TO: <", if it finds one, then it also adds decoded e-mail address and ends with ">\r\n".

7. Which process(es) does this malware attack and why?

Explained at answer 6.

8. What is the significance of the .ini file?

The .ini file is encoded e-mail address that all of the sent e-mails will be also directed to.

In our given example, there is encoded string "CHMMXaL@MV@SD@O@MXRHRCNNJBNL", to check what it decodes to, we should set a breakpoint in a debugger after returning from decoding function.

The call to decoding function is at 0x100016CA and the decoding function starts at 0x100010B3.

The encoded string translates to billy@malwareanalysisbook.com which is the additional recipient of all of the outgoing e-mails.