



Reversing Malware Functionalities

Downloader

- Downloader downloads additional malware and executes it
- Downloader uses **UrlDownloadToFile()** api to download the file from an URL and saves it onto the disk.
- It then calls **ShellExecute()** or **WinExec()** or **CreateProcess()** to execute the downloaded file.

Lab 7 - The case of Joon malware

Analyze the sample **blob.exe** and answer the following questions:

- Does the malware have references to the downloader API calls?
- What is the API call used by the malware to download the file?
- What is the name of the domain from where malware downloads malicious component?
- What is the name of the executable that it downloads from the domain?
- What is the full path on the disk where downloaded malware is dropped?
- How does it execute the downloaded file?
- Based on your analysis, what is the functionality of the malware?

Dropper

- Dropper comes with additional malware embedded in it and it then drops that piece of malware and executes it.
- Dropper normally embeds executable in the resource section, it can extract the embedded executable using below API's
 - **FindResource**
 - **LoadResource**
 - **SizeOfResource**
 - **LockResource**

Demo 8 - Reversing the JOON Malware

Keyloggers

Designed to log keystrokes

Two most common methods:

- Install hook for the keyboard events - In this case, the malware can install a filter function for the keyboard event using **SetWindowsHookEx()** and malware can also monitor the mouse event.
- Polls keyboard state with **GetAsyncKeyState()** - In this method, the malware can check if the key is pressed by calling the **GetAsyncKeyState()** function, the malware will run this function in a loop and periodically monitors for all possible keystrokes.

Keylogger using SetWindowsHookEx

```
.text:00401516      push    0                ; lpModuleName
.text:00401518      call   GetModuleHandleA
.text:0040151D      push    0                ; dwThreadId
.text:0040151F      push    eax              ; hmod
.text:00401520      push    offset hook_proc ; lpfn
.text:00401525      push    WH_KEYBOARD_LL   ; idHook
.text:00401527      call   SetWindowsHookExA
.text:0040152C      mov     ds:hkh, eax
.text:00401531      test   eax, eax
.text:00401533      jnz    short loc_40156A
```

Monitors the Keyboard events

Keylogger using GetAsyncKeyState()

The screenshot shows assembly code for a keylogger loop. A box labeled "Loop" points to the start of the code block. The code includes several instructions with annotations:

- `push VK_SHIFT` and `call GetKeyState`: ; CODE XREF: StartAddress+419↓j ; nVirtKey. Annotation: Determines if shift key is pressed.
- `mov edi, eax`, `movsx edi, di`, `mov [ebp+var_8], edi`, `mov edi, [ebp+var_4]`: Preparation for the next key check.
- `mov ebx, vKey_codes[edi*4]`, `push ebx`, `call GetAsyncKeyState`: ; vKey. Annotation: Determines if a particular key is pressed.
- `mov edi, eax`, `movsx edi, di`, `test di, 8000h`, `jz short loc_4014E8`: Annotation: Checks if the most significant bit is set to determine if key is pressed.
- `push VK_CAPITAL`, `call GetKeyState`: ; nVirtKey. Annotation: Determines if CAPS LOCK is on.
- `or ax, ax`, `jz short loc_401496`: End of the key state checks.

The screenshot shows the continuation of the assembly code, specifically the loop increment and comparison instructions:

- `inc [ebp+var_4]`: ; StartAddress+1EA↑j ...
- `cmp [ebp+var_4], 5Ch`: ←
- `jnl loc_40143F`: Loop back to the start of the keylogger loop.

Demo 9 - Analysis of Keylogger

HTTP backdoor

Takes command via http or https

Use of http or https bypasses the firewall rules and easy to blend in with normal user traffic

Steps:

- Creates http connection using **InternetOpen()** or **InternetConnect()**
- Build http request using **HttpOpenRequest()**, **HttpAddRequestHeader()**
- Send http request using **HttpSendRequest()**
- Read response using **InternetReadFile()**

Demo 10 - Reverse Engineering APT1 Malware