

History

Version 1.0 : 17th June 2006

Plugin for version PESpin 1.3 & 1.304 is released.

PESpin & ImpREC

PESpin is a Windows executable files compressor & protector. It has some of the best protection techniques to protect the binary against disassembling and debugging. When you try to unpack any PESpin protected file, you will experience several obstacles and one of them is API redirection. This cannot be circumvented by just using ImpREC as PESpin uses API redirection in combination with API emulation technique.

ImpREC is the popular tool in the reverse engineering world. It is a most powerful import rebuilder known for its amazing features. It supports various mechanisms to trace the API calls and also allows the users to write their own custom plugins. It comes bundled with several plugins for tracing binaries protected with teLock, ASProtect etc. This article will explain on how to write such a plugin for tracing API calls for PESpin and how one can break its protection against reconstructing import table. It can also serve as guide to write the API tracer plugin for any such protection software.

PESpin's API Redirection mechanism

Before we proceed to write a plugin we have to understand how the API redirection is implemented in PESpin. To start with, protect any binary file with PESpin. I have used my LDAPSearch application which you can find [here](#).

Here I will explain only the portion of unpacking which is relevant to breaking API redirection technique. Now launch the target application in Ollydbg and make it to break on OEP. Once you are at the OEP, your next job is to rebuild the import table. Launch ImpREC and select the LDAPSearch application from the drop down list. Enter the current OEP and press "IAT Autosearch" button. ImpREC kindly reports that it could not find anything. This is due to API Redirection technique used by PESpin. There is one more way to get the API calls. Now on the ImpREC right click and select "Get API Calls" from "Advanced Commands" and then click 'OK' on the new dialog box. It will list some of the valid DLL entries along with many invalid entries. Lets check what evil this PESpin is doing with import table.

Some of the invalid entries lies in the address range 18000 - 19000. These are really INVALID ones. You can make out if you are into this unpacking business since quite long time. Next entries starts with address 25***. Lets consider the first one which is at address 0x2542A. Click on that node and you observe that its pointing to address 0x880000. Now go back to OllyDbg and start disassembling at the address 0x880000. There you will find the instructions listed below.

```
00880000 EB 01 JMP SHORT 00880003
00880002 D957 8B FST DWORD PTR DS:[EDI-75]
```

Just follow the JMP instruction you will arrive here.

```
00880003 57 PUSH EDI
00880004 8B7C24 08 MOV EDI,DWORD PTR SS:[ESP+8]
00880008 81FF 60EA0000 CMP EDI,0EA60
0088000E EB 07 JMP SHORT 00880017
00880010 FFE9 JMP FAR ECX ; Illegal use of register
```

These are actually first few instructions copied from the actual API function. Next follow the JMP instruction at 0x0088000E. You will see this.

```
00880017 EB F8 JMP SHORT 00880011
00880019 EB 01 JMP SHORT 0088001C
```

Now follow the first jmp instruction and you will arrive here finally.

```
00880011 -E9 94845F77 JMP kernel32.77E784AA
00880016 EA EBF8EB01 D9A1 JMP FAR AlD9:01EBF8EB ; Far jump
```

This code is jumping to one of the kernel32 function. Lets follow it and once you are in kernel32 just scroll above and look at those instructions. You will find that those instructions exactly matches the instruction we have just encountered. So the function actually begins at address 77E7849F and looking at export table of kernel32 you can find that this address points to the function GetCPIInfo.

With this information in hand lets see how can we write a ImpREC plugin for automating this process.

How to write plugin?

Writing plugin for ImpREC is simple and straight forward. You can have look at plugin samples bundled with it to get better idea. To start with you need to create a DLL with empty DLLMain() function and must export function called Trace(...) whose prototype is given below.

```
DLLEXPORT DWORD Trace(DWORD hFileMap, DWORD dwSizeMap, DWORD dwTimeOut, DWORD dwToTrace, DWORD dwExactCall);
```

Parameters:

```
-----
hFileMap : HANDLE of the mapped file used for returning actual API address found.
dwSizeMap : Size of that mapped file
dwTimeOut : TimeOut of ImpREC in Options
dwToTrace : Redirected API address for which we have to find actual API address.
dwExactCall : EIP of the exact call (in VA)
```

Returned value:

On success return the value 200. Otherwise returns any value above 200.

During actual operation while tracing, this DLL will be injected into target process and the trace code will be executed in the context of target process. For each address, ImpREC calls the function 'Trace' with dwToTrace value set to the address to be resolved. In our case this address is 0x880000 and the return address will be 0x77E7849F. That's all we have to do.

Now we know how the PESPIn has implemented API redirection. Lets summarize the steps to get actual API address.

- First instruction is EB 01. To skip this just add 3 to the starting address.
- Next go through each byte until you encounter EB 07 instruction. During this loop keep the count of bytes.
- Once you get EB 07 instruction, add 3 to current address to reach jmp <real api> instruction.
- Final address will be calculated as follows
address = <jmp api addr> + <next instrn addr> - count of emulated bytes.

Now we have enough information to complete the Trace function. Lets start coding...

1) Map the file

First map the view of the file in write mode. Note that ImpREC uses memory mapped file to get back the API address from the Trace() function. Once you find the address you have to write it to the beginning of this mapped file.

```
DWORD* dwPtrOutput = (DWORD*)MapViewOfFile((HANDLE)hFileMap, FILE_MAP_READ | FILE_MAP_WRITE, 0, 0, 0);
```

2) Trace the address

We have to just convert above mentioned steps into computer code. The tracing code is given below.

```
BYTE *taddr = (BYTE*)dwToTrace;
taddr = taddr + 3;

int byteCount = 0;
while(1)
{
    // loop until EB 07 instruction comes
    if( taddr[0] == 0xEB && taddr[1] == 0x07)
        break;

    byteCount++;
    taddr++;
}

taddr = taddr + 3;
```

3) Compute final API address

Now we are at the jmp <real API address> instruction. Note that this jumps to middle of the function as some of the instructions from this function are already emulated. To get exact address we have to subtract the count of emulated bytes from this address. Also since this is relative jump we have to add the address of next instruction to the above address.

```
jmpAddress = *((DWORD *) (taddr + 1));
nextAddress = (DWORD) (taddr + 5);

finalAddress = jmpAddress + nextAddress - byteCount;
```

4) Return the API address

Now we have to store this address at the beginning of mapped file and return the success value 200.

```
*dwPtrOutput = finalAddress;
UnmapViewOfFile((LPCVOID)dwPtrOutput);
CloseHandle((HANDLE)hFileMap);

return (200);
```

Testing the Plugin

Now build the plugin DLL and put into ImpREC's plugin directory. Next launch the ImpREC again and then select the target process from the drop down list. Now right click on the panel, select "Get API Calls" from "Advanced Commands" and then click OK on the new dialog box. It will list some of the valid DLL entries along with many invalid entries. Now click on "Show Invalid" button and all the invalid entries will get highlighted. Next right click, select plugin tracers and then select our PESpin plugin from the list. ImpREC will happily display the APIs as they are resolved one by one.

Well, we have finished with our first PESpin plugin for ImpREC. Now you have to just delete remaining invalid ones and continue with rest of unpacking process.

Reference

- PESpin : Windows PE file protector
- ImpREC 1.6 : Import table reconstructor for Windows PE files.

For any queries and suggestions, please drop a mail to [tnagareshwar at gmail.com](mailto:tnagareshwar@gmail.com)
