Compressing the Imports Section

Download sample code

In the first article, we dicsussed how to build a simple executable crypter and walked you through the basic steps on its design and implementation.

In this article, we are going to move on to the second stage of building an executable crypter, compressing/modifying the images Imports table.

When the system loader initalizes a executable, one of the steps it takes is to load all necessary dlls and place pointers to the functions it uses into a table where the executable can use them.

The Import Table holds the: dll names, function names, and base address for the function pointer table; While the Import Address Table (IAT) holds the actual function pointers used to access each of the functions.

The reason only the IAT base address (first thunk) is needed in the Import table data is that each function pointer is simply a 4 byte number. So the first function address is at first_thunk, the next at first_thunk+4 etc (numbers in byte offsets not int array offsets)

So, from here, you know what functions an exe is going to import, and what offsets the function pointers must reside at. Now we can devise a way to remove them from the actual executable, and make some custom code of our own to restore them at runtime.

Why would we do this? Well you probably wouldnt have found this paper if you have that question...So lets just assume its neat, its cool, its great, like cool aid but without the fat guy running though walls.

So where do we start. First, what information will we need to restore IAT

```
a) Dll name
b) First Thunk
c) List of function names to import
```

Ok that is not so bad. How about implementation what should we know.

Well first off, we need to hold our data in a format which will easily able to load multiple dlls, and multiple function names. The format i used for this demo is very simple. I did not compress/decompress them for readability sake, but compression would be very easy at this point as it is already extracted to data string.

Next thing to think of is how to run our custom code on startup. Since this is a trainer, I have chosen easiest method which is to build a dll and place all of the IAT restoration code in its DLLMAIN. When we modify the target executable to remove all of the real imports, we will also add our own dll to its import table so that it is automatically loaded by the windows loader.

For basic review of example loader implementation:

- uses 3 parallel arrays. one for dllname, 1st thunk, and api fx name list (a-c above)
- for loop loads each dll, sets pointer for 1st thunk, then parses functionname list calling GetProcAddress on each.
- Function address is then placed in correct spot in IAT, and thunk pointer incremented to move it the next spot.

See, when you really lay it out, its not as bad as it may first seem. So thats enough detail on the code. Read through it , see comments, its not that hard I think it is less than a page of C.

Last thing left is to modify import table for target exe. First open it in LordPE goto directory and click ... to edit Import table. Right Click on first entry, and choose add import. Here we just enter our new dll name and its one dummy exported entry. This export will never be called just there so windows loader loads dll.

Now you can click on kernel32.dll and kill it. You may also want to hexedit exe to remove api name lists for whatever its worth.

Now when exe loads, windows loader will load our dll and out dllmain will execute before main entry point of program. This gives our code time to run and restore IAT to how application expects it.

Now that you have the basics you can make a more realistic work by writing code to:

- Add in registration checks and trial timeouts.
- Add anti debugging/dumping code
- Dynamically scan exe and build Import data lists
- Embed IAT rebuilder right into exe
- Build in IAT redirection to hose up debugging/dumping (example included)
- etc.

Tons of tricks andextra work are possible. With basics down now rest is not as far off.

Download sample code

-dzzie <dzzie at yahoo.com>