

BEFORE READING THIS MANUAL PLEASE NOTE THIS IS A BETA VERSION. DEVELOPMENT OF IMPORTS FIXER HAS BEEN DONE OVER MANY YEARS WHILST THIS HELPFIELD HAS SEEN LITTLE IF NO CHANGES AND WAS NEVER INTENDED FOR PUBLIC RELEASE. CERTAIN FEATURES AND FUNCTIONS DESCRIBED WILL HAVE BEEN REMOVED OR AS YET NOT IMPLEMENTED FOR THE PUBLIC BETA RELEASE. PLEASE DO NOT TAKE THE INFORMATION CONTAINED WITHIN THIS MANUAL WHOLLY OR PARTLY AS BEING ACCURATE, THERE WILL BE MANY ERRORS. A COMPLETE DOCUMENT REWRITE WITH MORE DETAIL AND MATERIAL ACCURACY WILL BE RELEASED ALONGSIDE THE FINAL PUBLIC BUILD OF IMPORTS FIXER.

## INTRODUCTION AND PURPOSE

**Imports Fixer** (abbreviated to **IF** hereafter) has been specifically created to assist in the process of rebuilding and reconstructing portable executable files found in memory. IF has been designed to rebuild imports for Win32 Portable Executable and Dynamic Link Libraries (DLL's). With IF one can dump a "running" executable to disk even after cutting away unwanted sections or after including the allocated memory blocks of your choice in the dump (which is very useful when dealing with redirected API's). IF allows you to easily reconstruct a new Image Import Descriptor (IID), Import Array Table (IAT) with ASCII modules and function names. IF can rebuild section tables even in the case of cut sections or allocated memory blocks dumped as new sections. With IF you can edit the Optional Header Data and edit the sections.

Imports Fixer was primarily designed to assist in manual unpacking process and as development continued more features were added; Converter Tool, Hex Calculator and IF Express.



## FEATURES OF IMPORTS FIXER

## ▶▶▶Dumper

- Built-in dumper
- Support for dumping redirection code

## Fix Imports

- An original tree view
- 2 different methods to find original imports (by IAT and/or API calls)
- A complete imports rebuilder (including a new fresh IAT)

## ▶▶ Tracers

- 3 default tracers (disassembler, hook & ring3) to find APIs in redirected code
- A plugin interface to support third-party tracers

## ▶▶▶ Rebuild

- Rebuild the dumped file to max compression
- **Support for dumped allocated memory blocks**

## ►►PE Editing

- To be adjusted when feature is implemented

## ►►Other

- Support ALL 32bits Windows (9x, ME, NT, 2k, 2k3 and XP)
- An export renormalize for Win9x/ME (ala Icedump)
- A built-in coloured disasm/hex-viewer to analyze the redirected code
- Support for almost all known antidump tricks



## JUMPSTART

Are you an experienced user wishing to jump right in? Just follow these few points

...

IMPORTANT: throughout this whole Help file and to avoid every misunderstanding, with the following terms is meant:

Dumping: the dumping from memory to disk from a "running" file.

Fixing: the reconstruction from the imports (Image Import Descriptors, Import Array Table and the ASCII modules and function names).

Rebuilding: the reconstruction from the section table (array of IMAGE\_SECTION\_HEADER's structures)

### **1. Will I be able just following these few steps below to dump/fix/rebuild a packed/protected executable successfully?**

Just follow these few steps below and see if you can find your way with it. However, for more advanced dumping/fixing/rebuilding, I strongly advice to study this complete helpfile, especially in those matters where Imports Fixer offers more and/or extended possibilities in comparison to the existing tools.

### **2. So, I have a compressed/protected executable. How do I start ?**

If you want to use Imports Fixer, than a basic knowledge of unpacking/de-protecting the targeted packer/protector is assumed. It is assumed that you have knowledge of the Original Entry Point (OEP) of the original (NOT packed) target.

### **3. Ok. I have knowledge of OEP. Now what?**

First step is to have the target paused (suspended after unpacking/de-protecting) at OEP. This is only very occasionally NOT necessary in which cases you can just attach to the running exe, dump and change to the new OEP on disk. Now, you'll have to dump the target to disk. For this, run IF, attach to the "running" target (and/or a DLL), right-click and choose "Dump". Proceed with dumping (Dumper Tool).

Remark: check the right options for dumping on the "Preferences" tab (if changes are necessary).

### **4. I dumped the target successfully. Am I done yet?**

That depends. In some cases your job will end here. Most of the time however,

you'll need to fix imports, perhaps even rebuild the section table and/or edit some (optional) header data.

### **5. Indeed, the dumped exe doesn't run. What now?**

I'm assuming some knowledge here from your part. But most likely, you'll need to fix imports. Hence, proceed to the "IT/IAT" tab and fix the target's imports. Remark: check the right options for fixing on the "Preferences" tab (if changes are necessary).

### **6. Oops ... most of the dumps run fine, but this one doesn't! Have I made a mistake?**

Perhaps not! Especially if you've cut some compressor sections (which are now useless) or added some allocated memory blocks as sections: you'll still need to rebuild the section table! Perhaps you may also need to edit some data in the optional header for smooth running exe's. Proceed to the "Rebuilding" tab and rebuild the target. Remark: check the right options for fixing on the "Preferences" tab (if changes are necessary).

### **7. What is that about editing data in the optional header for smooth running exe's?**

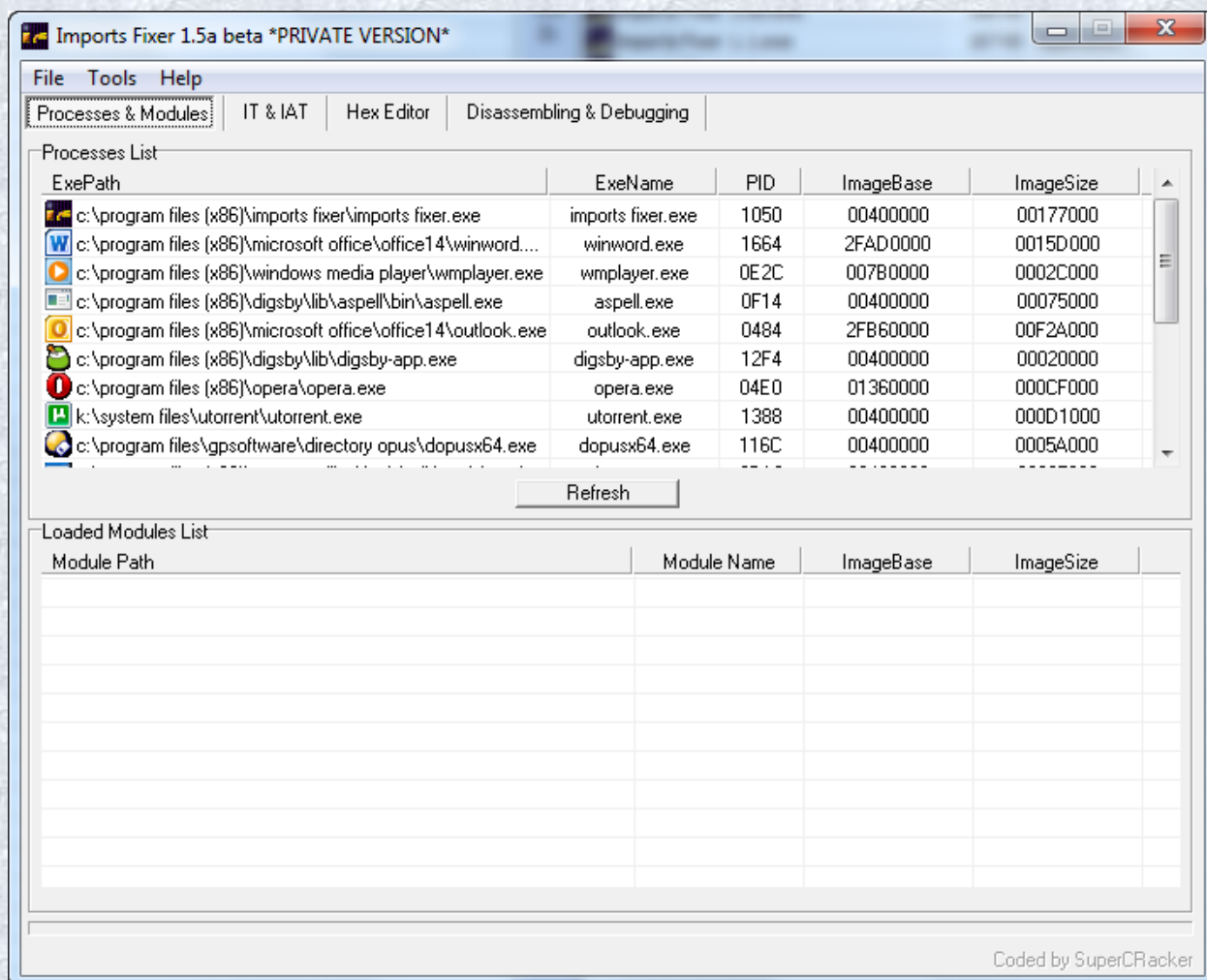
Some packers/protectors edit or even destroy the header. Minor editing in the optional header data can be achieved from the PE Editor tab. Proceed to the "PE Editor" tab and make the necessary changes. Don't forget to save your work! Remark: to replace by a "new" complete header from a file on disk -> see the "Preferences" tab. **This option should be on this PE Editing tab, not on the preferences!**

**This page needs adjusting as new features get implemented**



## **GENERAL OVERVIEW**

Imports Fixer is a tool to easily **dump**, **fix** and **rebuild** a "running" executable. Already at its startup, Imports Fixer automatically detects all running processes and displays these in the Process List. Choosing (by highlighting) a process, automatically displays all loaded modules (dll's) from this process. Notice that the program is built as a tabbed layout (Processes & Modules | IT & IAT).



## Requirements

- A running process of your target
- Your knowledge about the Original Entry Point (OEP), or a valid OEP elsewhere (redirection code).

## QUICK FUNCTION OVERVIEW

You may click the "Refresh" button if desired after Imports Fixer having automatically detected all running processes and having displayed these in the process list.

## How to proceed?

- 1 - Select the target in the "Process List". Imports Fixer will automatically read the

process and display all its loaded modules in the "Loaded modules List". The target's Entry Point (EP) will be automatically written in the OEP editbox. If you want to address a dll from the running process, simply highlight it from the "Loaded Modules List".

- \* If your real target is a dll, choose the [DLL](#) to select it.

- \* You can dump your target with the right-click popup menu correctly IF AND ONLY IF you have suspended your target at OEP.

**2** - You may decide to dump the file at this point (if fixing is NOT necessary): see "dumping".

**3** - If you have knowledge about the OEP's address : enter it in the appropriate editbox ([IAT](#) Relative Virtual Address) with its Relative Virtual Address value ([RVA](#)) and press "IAT AutoSearch" button to get possible IAT RVA address and IAT Size values which might contain the original IAT. If you don't have knowledge about the OEP's address yet, it will be imperative to find the OEP first, then enter the RVA and size of IAT following your manually findings.

**4** - Press "Get Imports" button if the "IAT AutoSearch" has found something. Some compilers can produce a non-contiguous IAT, hence the "IAT AutoSearch" function may not find the complete IAT. For this case, enter the RVA and Size of the whole section which contains that IAT (see the messagebox containing that info just after clicking "Get Imports").

Remark : The IAT could be completely (or partially) removed by the packer/protector so that the "IAT AutoSearch" will fail. In this case, use the "Get API Call" command (right click on the tree). It will add to your imports, all pointers <XXX> of all <CALL [XXX]> and <JMP [XXX]> directly in the code. Do not forget to precise all sections which contain or are supposed to contain code (usually it is only the first section so you have nothing to do because it is the default configuration). To precise these sections, right click on the tree for a "Select Code Section(s)". This method is based on heuristics so you might have to analyze and remove by hand all invalid slots.

**5** - Use the "Show Invalid" button to see all unresolved slots. You will need to trace into them to find the real API. Always try the Tracer Level1 first because it does not execute any code of the target. The Tracer Level2 is to be used in the last resort because it is the less stable one (it uses a global hook).

- \* If you need to retrace into an already traced slot, you have to invalidate it (right click on it and use "Invalidate function(s)" command).

- \* Use "Cut thunk(s)" to remove individually each function

- \* Use "Delete thunk(s)" to remove individually each module

- \* Double click on a slot to edit it manually if you know the real [API](#).

**6** - Use the "Show Suspect" button to check 'supposed' wrong traced slot. This is possible because of the Tracer Level1 for example.

- \* A suspect slot is an alone valid function in a thunk or an API which already exists in the same thunk (ie several same apis in the same module)

- \* A suspect slot is not necessary invalid. It only needs a quick analyze of your part.

**7** - If you do not want to add a new section and know where you can put the new rebuilt import (in the last section for example), uncheck "Add new section" and enter the wanted RVA. The easiest way is to add a new section though (by default).

**8** - Press "Fix Dump" to fix your DUMPED file. You do not need to make a backup. If your filename is "Dump.exe", it will create "Dump\_IF.exe". Moreover the EP of your dump will be fixed to the value you have entered if you turned "Fix EP to OEP" on, in Options.

You will also have these options:

- \* "Auto reloc": Normal mode for relocations. It will reloc only the ripped region by tracing instruction per instruction (with the disasm engine)
- \* "Hardcore reloc": Hardcore mode for relocations. It will reloc the whole region in addition to the "Auto reloc"
- \* "Rebuild Imports": All imports in the regions will be rebuilt. It means Imports Fixer will stick to the current imports, the imports needed by the ripped code.

## **DUMPER TOOL**

Remark: throughout this whole Helpfile and to avoid every misunderstanding, by "Dumping" is meant the dumping from memory to disk from a "running" file.

Right-clicking the desired process in the "Process List" brings up the Dumper Tool popup menu.

| Processes List                                                                                                                                   |                   |      |           |           |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------|-----------|-----------|--|
| ExePath                                                                                                                                          | ExeName           | PID  | ImageBase | ImageSize |  |
|  c:\program files (x86)\imports fixer\imports fixer.exe       | imports fixer.exe | 1050 | 00400000  | 00177000  |  |
|  c:\program files (x86)\microsoft office\office14\winword...  | winword.exe       | 1664 | 2FAD0000  | 0015D000  |  |
|  c:\program files (x86)\windows media player\wmplayer.exe     | wmplayer.exe      | 0E2C | 007B0000  | 0002C000  |  |
|  c:\program files (x86)\digsby\lib\aspell\bin\aspell.exe      | aspell.exe        | 0F14 | 00400000  | 00075000  |  |
|  c:\program files (x86)\microsoft office\office14\outlook.exe | outlook.exe       | 0484 | 2FB60000  | 00F2A000  |  |
|  c:\program files (x86)\digsby\lib\digsby-app.exe             | digsby-app.exe    | 12F4 | 00400000  | 00020000  |  |
|  c:\program files (x86)\opera\opera.exe                       | opera.exe         | 04E0 | 01360000  | 000CF000  |  |
|  k:\system files\utorrent\utorrent.exe                        | utorrent.exe      | 1388 | 00400000  | 000D1000  |  |
|  c:\program files\gpssoftware\directory opus\dopusx64.exe     | dopusx64.exe      | 116C | 00400000  | 0005A000  |  |
| Refresh                                                                                                                                          |                   |      |           |           |  |

Left-click "Dump" to bring up the Dumper Tool.

**Dumper Tool**

**General Info**

Start Address :  Size of Dump :

Entry Point :  Modify Entry Point to :   Not supported yet

**Sections Overview**

| Name                                       | VOffset  | VSize    | ROffset  | RSize    | Flags    |
|--------------------------------------------|----------|----------|----------|----------|----------|
| <input checked="" type="checkbox"/> .text  | 00001000 | 00003000 | 00001000 | 00003000 | 60000020 |
| <input checked="" type="checkbox"/> .rdata | 00004000 | 00001000 | 00004000 | 00001000 | 40000040 |
| <input checked="" type="checkbox"/> .data  | 00005000 | 00002000 | 00005000 | 00002000 | C0000040 |
| <input checked="" type="checkbox"/> .rsrc  | 00007000 | 00019000 | 00007000 | 00019000 | 40000040 |

N. of original sections : 4      N. of extra sections added : 0      Max N. of extra sections to add : 86

☒ Check/Uncheck original sections

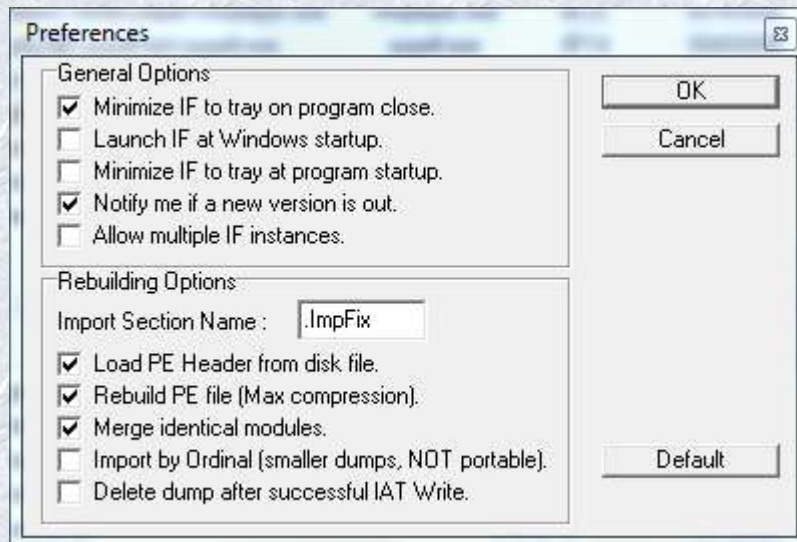
☒ Fix Dump Image ROffset and RSize      ☒ Close Dumper Tool after a successful dump 

Check any sections **and any allocated memory blocks** that you want to dump. The executable's sections are checked by default (full dump). Uncheck the ones not needed for a smaller dump (ie. compressor sections). **Decide at this point if you want to rebuild the dumped file now or when fixing (or not at all).** Click the "Dump" button to dump after correcting the general info box values if needed or click the "OK" button to leave without dumping.

## **REBUILDING OPTIONS**

Remark: throughout this whole Helpfile and to avoid every misunderstanding, with Rebuilding is meant the reconstruction from the section table (array of IMAGE\_SECTION\_HEADER's structures)

The "Rebuild PE File" feature (**see updatelist : may possibly be implemented by SC as a button to give a choice to rebuild when dumping or to rebuild when fixing an IAT**) rebuilds any dumped file with maximum compression **at any time**. This feature can be set in the preferences menu: Menu --> Tools --> Preferences. Just check the checkbox (**or/and navigate to the dump you wish to rebuild if SC made a button for it**). (In case SC makes a rebuilding option also for dumps including dumped allocated memory blocks: Full compatibility is reached, also for dumps that contain other blocks of code outside the executable's sections).



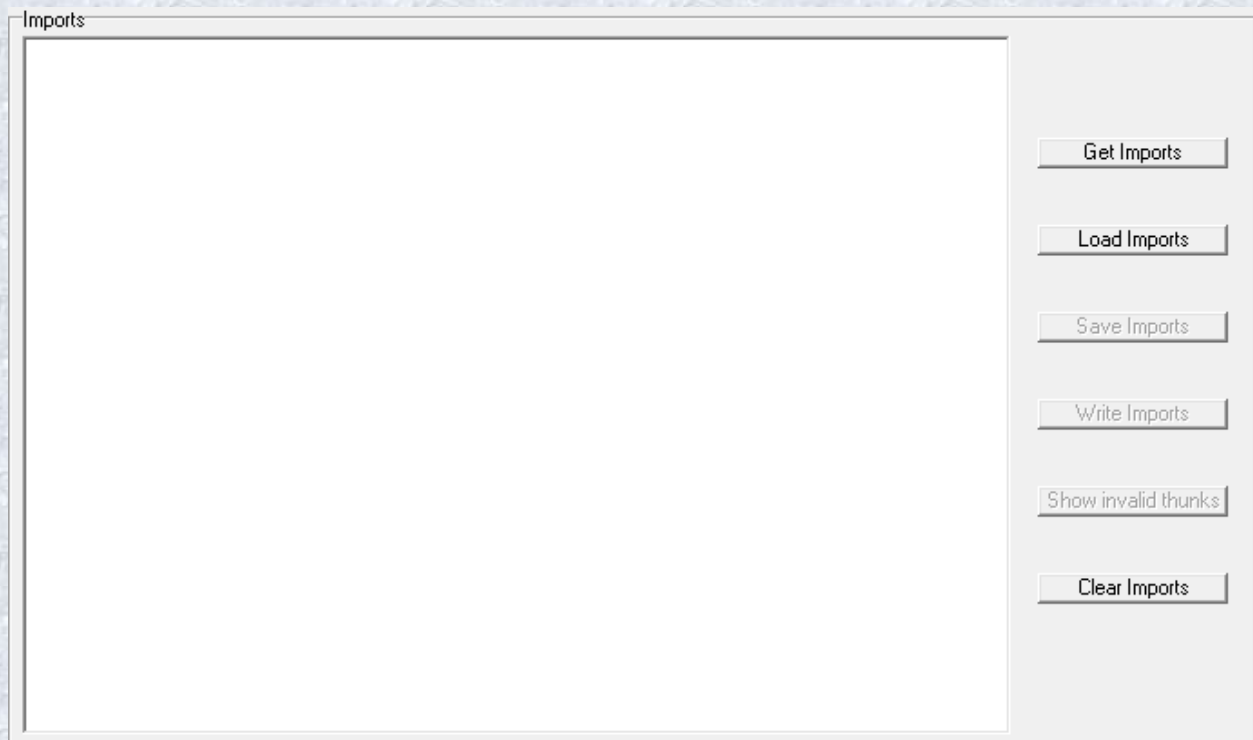
## REPAIRING / REBUILDING IMPORTS

Remark: throughout this whole Helpfile and to avoid every misunderstanding, with "Fixing" or "Fix Imports" is meant the reconstruction from the imports (Image Import Descriptors, Import Array Table and the ASCII modules and function names).

**1** - Enter OEP (following your manual findings) in the appropriate editbox and press "IAT AutoSearch" button to get a possible IAT RVA address and IAT Size values which can contain the original IAT. Else: enter the RVA and size of IAT following your manual findings.

| IAT Information |                                       |           |                                                |
|-----------------|---------------------------------------|-----------|------------------------------------------------|
| OEP:            | <input type="text" value="00002C61"/> | IAT RVA:  | <input type="text" value="00000000"/>          |
|                 |                                       | IAT Size: | <input type="text" value="00001000"/>          |
|                 |                                       |           | <input type="button" value="IAT Auto Search"/> |

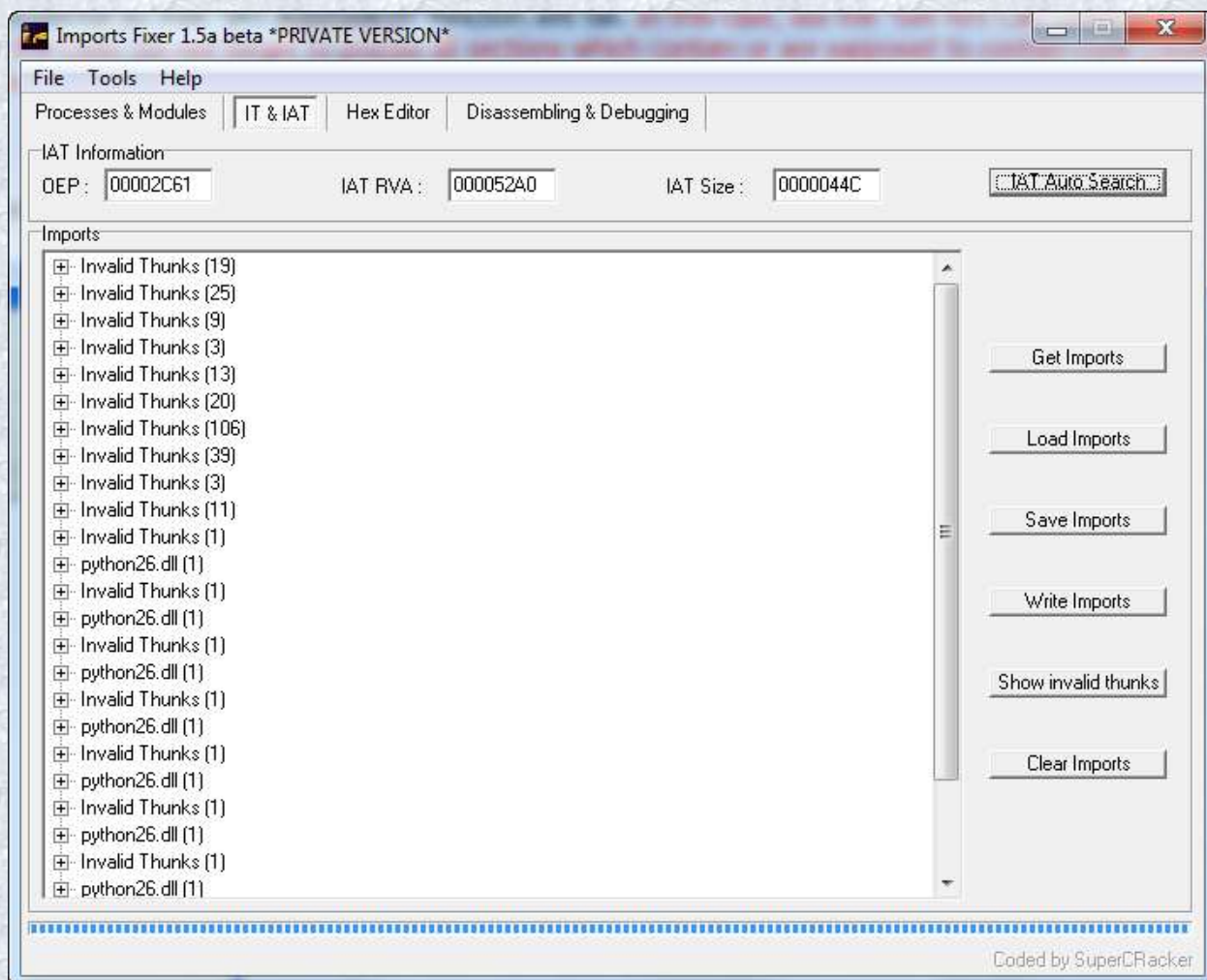
**2** - Press "Get Imports" button if the "IAT AutoSearch" has found something or after your manual input. For Borland targets (for example), "IAT AutoSearch" may not have found the complete IAT. It is because these targets do not have a contiguous IAT. For this case, enter the RVA and Size of the whole section containing that IAT (that information is ALWAYS written in the messagebox and also in the Log window just after clicking "Get Imports").



#### NOTE:

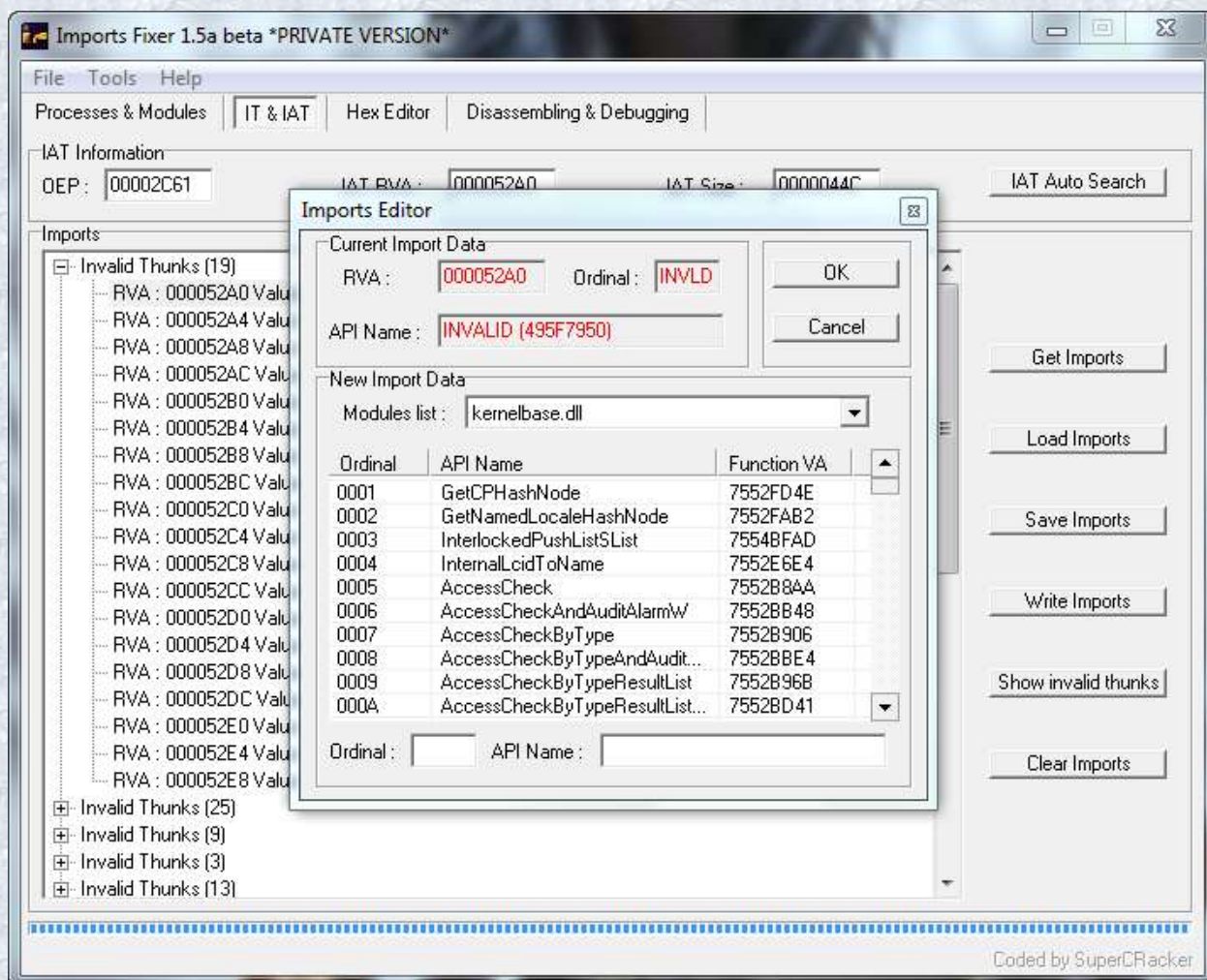
-----  
The IAT could be completely (or partially) removed by the packer/protector so that "IAT AutoSearch" function will fail. In this case, use the "Get API Call" command (right click on the tree). It will add to your imports, all pointers <XXX> of all <CALL [XXX]> and <JMP [XXX]> directly in the code. Do not forget to precise all sections which contain or are supposed to contain code (usually it is only the first section so you have nothing to do because it is the default configuration). To precise these sections, right click on the tree for a "Select Code Section(s)". This method is based on heuristics so you might have to analyse and remove by hand all invalid slots.

**3** - The imports are found but there may be invalid ones (see picture).



**4** - Use the "Show Invalid" button to see all unresolved slots. **You will need to trace into them to find the real API. Always try the Tracer Level1 first because it does not execute any code of the target. The Tracer Level2 is to be used in the last resort because it is the less stable one (it uses a global hook).**

- ▶ If you need to retrace into an already traced slot, you have to invalidate it (right click on it and use "Invalidate function(s)" command).
- ▶ Use "Cut thunk(s)" to remove individually each function
- ▶ Use "Delete thunk(s)" to remove individually each module
- ▶ Double click on a slot to edit it manually if you know the real API



**5 - Use the "Show Suspect" button to check 'supposed' wrong traced slot. This is possible because of the Tracer Level1 for example.**

► A suspect slot is an alone valid function in a thunk or an API which already exists in the same thunk (i.e. several same apis in the same module)

► A suspect slot is not necessary invalid. It only needs a quick analyse of your part.

**6 - If you do not want to add a new section and know where you can put the new rebuilt import (in the last section for example), uncheck "Add new section" and enter the wanted RVA. The easiest way is to add a new section though (by default).**

**7 - Press "Write Imports" to fix your DUMPED file. You do not need to make a backup. If your filename is "Dump.exe", it will create "Dump\_IF.exe". Moreover the EP of your dump will be fixed to the value you have entered if you turned "Fix EP to OEP" on, in Options.**

**You will also have these options:**

► **"Auto reloc": Normal mode for relocations. It will reloc only the ripped region by**

tracing instruction per instruction (with the disasm engine)

- ▶ "Hardcore reloc": Hardcore mode for relocations. It will reloc the whole region in addition to the "Auto reloc"
- ▶ "Rebuild Imports": All imports in the regions will be rebuilt. It means Imports Fixer will stick to the current imports, the imports needed by the ripped code.

## **SECTION LABEL EDITING**

On the PE Editing tab, you find the possibility to manually edit and save the section table structures (array of IMAGE\_SECTION\_HEADER's structures).

This page will need adjusting ASA features are implemented

## **OPTIONAL HEADER EDITING**

On the PE Editing tab, you find the possibility to edit data in the Optional Header structures.

This page will need adjusting ASA features are implemented

## **HELP MENU / TOOLS**

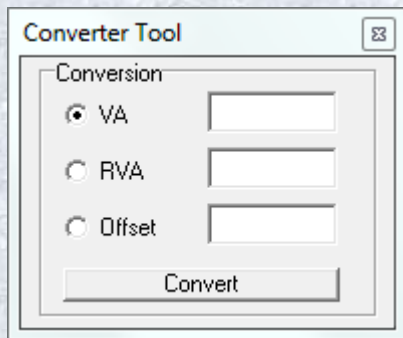
The Menu contains the side-functions in Imports Fixer because the main features are accessed either from the program's windows immediately or by pop-up menu's (right-click). The Menu is found on Imports Fixer's main screen and also allows access to some general purpose tools that aid in ease of use. The menu consists of the following:

### **- File menu:**

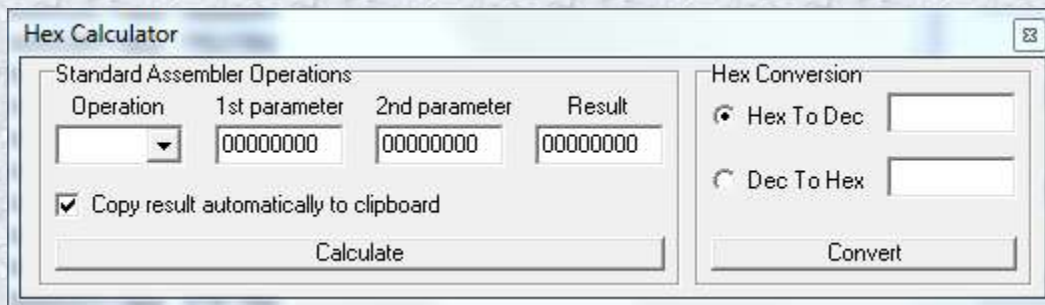
--> Exit: to exit Imports Fixer

### **- Tools menu:**

--> Converter tool: allows making conversions between Virtual Address (VA)/Relative Virtual Address (RVA)/Offset



--> Hex Calculator: allows calculating all standard assembler calculations and HEX/DEC/HEX conversion



--> IF Express: **will be filled in later**

--> Preferences: all preferences (options) relational to Imports Fixer. See Preferences for complete explaining.

## - Help menu:

--> Documentation: **calls (and refers to)** this Imports Fixer Helpfile

--> Check for updates ...: to update to the latest version of Imports Fixer

--> History: coding/debugging/updates history of Imports Fixer

--> About: all you may want to know about Imports Fixer and its author

## **USER COMMANDS**

Remark: also try rightclicking for certain options.

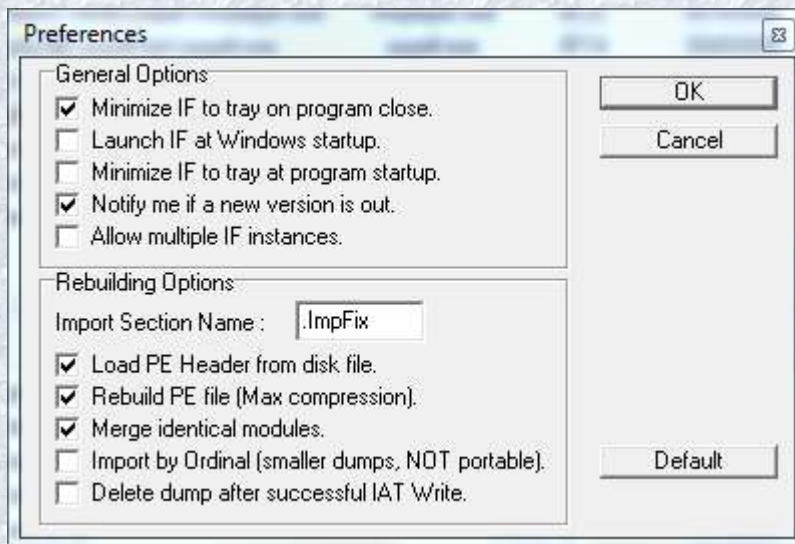
- "**Rebuild PE File**": Will automatically also rebuild the fixed exe. Especially useful when also redirected code is dumped

- "**Merge identical modules**": Will merge the identical modules but be cautious as this is sometimes NOT compatible due to the compiler and the way it handles information from the IAT. Use only if there are identical modules.
- "**IAT AutoSearch**": Input an OEP and this function will try to find the IAT RVA and Size. Study the MessageBox text well! On failure, you may have to find this manually yourself (FF 25 etc ...).
- "**Get Imports**": Build the IAT described by RVA and Size into a tree.
- "**Show Invalid**": Select all pointers in the IAT which are still invalid (redirected)
- "**Show Suspect**" : Select all valid functions which need a manual check of your part. It will select all same functions in the same thunk and all thunk which has only one function.
- "**Auto Trace**": An automatic tracer which will use the tracer level1 on all invalid pointers. It will apply the tracer level3 on suspect entries. Do not expect so much on it because it will never be able to replace some manual actions of your part.
- "**Clear Imports**": Clear all current imports
- "**Clear Log**": Clear the log window
- "**Load Tree**": Load a tree. This function allows to load a previously saved import tree again.
- "**Save Tree**": Save your current tree in a file. (text format)
- "**Fix Dump**": Add your current imports into the dumped file of your target
- "**Invalidate function(s)**": Invalidate a valid function. Use it if you know the current API is not valid and want to retrace it.
- "**Cut thunk(s)**": Remove all selected functions
- "**Delete thunk(s)**": Remove the modules of all selected functions which they belong to.
- "**Get API Calls**": This feature will get all call[] and jmp[] from the target to your imports. Its purpose is to find internal calls to the protector that does not belong to the IAT.
- "**Load PE Header**": To force Imports Fixer to use a specific PE Header. Can be useful in certain cases. This command WILL NOT PASTE the PE Header to your dump.
- "**Select Code Section(s)**": Used for "Get API Calls", "Create New IAT" and "Exact

Call". It tells them which sections contain code (to redirect all calls to the new IAT or to get all calls).

- "**Show Exact Call Window**": Show a window to see all the current "Exact Calls" found. You can remove them if needed (right click on the window).
- "**Get Imports Filter**": This command will set the range of the "valid" addresses to take care about. All the others will be removed.
- "**Disassemble / HexView**": Allow you to disassemble/hex-view the redirected code. Use right clicks to see possible functions on the disassembled code (address, imports...)

## **PREFERENCE SETTINGS**



- "**Import by Ordinal**": It needs less size than the usual import (by name and ordinal) but it will not be portable. Use it **\*\* IF AND ONLY IF \*\*** you want to rebuild the exe **ONLY** for your system and want a smaller file.
- "**Max Recursion**": The number of conditional jumps or calls the tracer level1 is allowed to follow. The higher this number, the longer it takes to finish: use it CAREFULLY. (But you can stop its progression by pressing "CONTROL+F12")
- "**Buffer Size**": is the size of the allocated memory block for tracing for EACH recursion. (Still for the tracer level1)
- "**Fix EP to OEP**": Turn on this option if you want to set the EP of your dump file to the OEP you have given in Imports Fixer.
- "**Time Out**": Time out for stopping tracers level2 and 3 in their progression. The value is in 50 milliseconds and in decimal.

- "**Enable Debug Privilege (NT/2K/XP)**": Enable this if your target is not listed in the process combobox selector. You will need to restart Imports Fixer to activate it.
- "**Fix Damaged PDB (Win9X/ME Only)**": Enable this if the target has hacked its PDB (ugly) against process opening. You will need to restart Imports Fixer to activate it.
- "**Use PE Header From Disk**": Use the PE Header of the disk rather than the dump.
- "**Renormalize Exports (W9X/ME)**": Use this to get a better portable executable. You must run Imports Fixer at least once before running your target with this option enabled. You will need to restart Imports Fixer to activate it.
- "**Exact Call**": Enable this option to rebuild schemes that use a unique pointer for several (or all) API redirections. BTW, an "Exact Call" is a reference in the code of a call to an API. Usually a call to an API is done through the IAT but some packers/protectors can modify directly the code to redirect to a unique pointer (this is also called "mangled scheme"). To fix an "Exact Call", Imports Fixer will need to fix the code to redirect it back to the IAT.
- "**Skip Main Slot**": Enable this if you want to trace only on all "Exact Calls" letting the main slot remained.
- "**TL3 Dumb Mode**": It has the behaviour of the Tracer Level1 but it will execute each instruction. Almost done for wrapped api which copies the start of the API and jumps "later".
- "**TL3 Log**": Enable this to log all EIP traced by the Tracer Level3. The max logged EIP can be set in the editbox near the checkbox. This will help you to debug where the tracer failed/stopped.
- "**Rebuild Original FT**": Enable this to also rebuild the original first thunks. This could help your disassembler to recognize imports better.
- "**Create New IAT**": Enable this option to create a new IAT.  
☐☐☐ This feature is needed when an "Exact Call" has no entry in the current IAT. You will be notified anyway if you did not enable it and try to fix your dump.  
☐☐☐ Do not forget to precise all sections which contain the code to be modified by this function, by right clicking on the tree for a "Select Code Section(s)"
- "**Mode Cloak**": Mainly for Anti-Imports Fixer methods, it could be used to all techniques trying to prevent ReadProcessMemory from another process. This option will change the name of the window and the API ReadProcessMemory will not further be used.  
☐☐☐ The "Full Dump" button will benefit of this option.  
☐☐☐ You can rename "Imports Fixer.exe" if the target tries to search Imports Fixer by process listing.

## IF PLUGIN SUPPORT

We'll see in this section a detailed explanation about plugins and how they can be implemented in IF.

### 1- What is a plugin?

---

A plugin is an external file (Dynamic Link Library) which can be used directly by the appropriate program "HOST". A plugin can be written in any available language and it is intended to bring extra features or to carry out additional tasks. To be recognised by the HOST, the plugin must respect a certain syntax which is imposed by the author of the HOST or even use a particular programming language. A plugin can contain only functions and procedures or it can include resources (like skins) or can be an independent DialogBox or sub-program into the big one (HOST).

### 2- Why not include directly plugins features in the HOST instead of using plugins?

---

-----

Plugins are very helpful and easy to use. Indeed, they are complementary to the HOST and can be used without installation. In a practical view, they make third-party coding possible and can be updated separately from the HOST. This way, the author doesn't have to update the whole release and lots of people will be contributing to the same program. Seeing that the author can't code everything in its HOST, it's an opportunity for other coders to implement new ideas and show their talent and creativity, the HOST will be then in permanent update.

### 3- Why plugins are used by IF?

---

As you know, IF has its own algorithm to trace, find and resolve IAT entries. Thus IF can resolve IAT functions in protected apps without IAT redirection or elimination. Unfortunately, this isn't sufficient due to the variety of protectors and IAT redirection tricks. Consequently, plugins will be used to complete the work of IF and resolve the remaining invalid thunks left unresolved by IF. In further versions, other types of plugins might be supported, I'm talking about IAT Auto Search plugins (for specific protected targets which IF fails to detect correctly the IAT location) and eventually IF skins.

### 4- What language must be used to write IF plugins?

---

You can use any language you're used to. The most important thing will be the syntax and the way the target is opened. Of course, some of you who have already written plugins for ImpRec will ask if their plugins are still operational in IF. Right

now it's impossible because ImpRec and IF are using different approaches while dealing with targets. IF uses the "ReadProcessMemory" API in its interventions whereas ImpRec uses the "MapViewOfFile" API. Also the parameters passed to Resolve function we'll be discussing in next part are different from ImpRec's one.

## 5- What is then the syntax to follow to write a compatible IF plugin?

---

You have to declare a function named "Resolve". You will have to respect the following syntax and parameters :

```
function  
Resolve(hProcess:DWORD;ImageBase:DWORD;Size:DWORD;RVA:DWORD):DWORD  
;
```

The previous declaration is written using Delphi, but there is no difference if you use another language. For example C declaration would be :

```
DWORD Resolve(DWORD hProcess, DWORD ImageBase, DWORD Size, DWORD  
RVA);
```

Now here's a little explanation of the parameters passed to this function :

- hProcess : It's the process handle of the target file. You will use the hProcess in ReadProcessMemory parameters.
- ImageBase : It's the ImageBase of the target. You will use this in ReadProcessMemory parameters.
- Size: It's the SizeOfImage in memory of the target. You will use this in ReadProcessMemory parameters.
- RVA: It's the RelativeVirtualAddress of the invalid thunk to resolve.

If all is OK the function return value must be the valid RVA of the API. Example: If the API is GetModuleHandleA the result should be its RVA, in my case: 77E59F93. It's IF job to extract the API name from its RVA. If there was an error during the resolve function the return value must be null (0).

Don't forget:

- To Allow virtual memory at the beginning and read the whole image of the target by using VirtualAlloc and ReadProcessMemory APIs.
- That you can get the size of a region of the target file by using the VirtualQueryEx API.
- That you can read any region of the target file even if it's outside the ImageBase+SizeOfImage range.
- To Free the allocated memory at the end by using the VirtualFree API.

For a concrete example, you can find the source code of teLock98 IF plugin in Samples subdirectory.

P.S: I have used an Assign function because I needed to alloc memory twice, the first one was for the whole memory image and the second was the region where IAT was redirected. Keep in mind that you aren't supposed to use such a function if you have to alloc memory once.

### **A summary for this part (Delphi example) :**

```
-----  
~~~~~  
~~~~~  
library MyPlugin;  
  
uses Windows;  
  
function  
Resolve(hProcess:DWORD;ImageBase:DWORD;Size:DWORD;RVA:DWORD):DWORD  
;StdCall;  
var hFile,NOBR:DWORD;  
begin  
hFile:=DWORD(VirtualAlloc(nil,Size,MEM_COMMIT,PAGE_EXECUTE_READWRITE));  
ReadProcessMemory(hProcess,Ptr(ImageBase),Ptr(hFile),Size,NOBR);  
// Your code here  
  
// Code end  
VirtualFree(Ptr(hFile),0,MEM_RELEASE);  
end;  
  
exports Resolve;  
  
begin  
end.  
~~~~~  
~~~~~
```

### **6- How can I contact you if I face a problem during my plugin coding?**

-----  
I hope that this little readme file has clarified some points concerning plugins and, of course, nothing is perfect. Problems, suggestions and criticism are always welcome, you can:

- Visit the official IF support website:  
<http://forum.tuts4you.com/index.php?showforum=119>
- Write an e-mail to the author at: [SuperCRacker3@hotmail.com](mailto:SuperCRacker3@hotmail.com)

## 7- Do I have the right to modify the content of this help file?

---

Absolutely not. Of course, this can't be done without my permission.

## 8- Greetings:

---

Thanks to all who supported me during IF programming especially Teddy Rogers and Lena151 from Seek n Destroy for testing and bug reports.

### **BUGS, REMARKS AND ADDITIONAL COMMENTS**

**Imports Fixer** was made with the greatest care and has been tested thoroughly. However, it's still under continued development. Due to its nature, there are probably still some bugs remaining. It would be helpful if these could be reported in IF support forum or to my email address with a clear explaining what exactly happens when and why. A good help could also be to provide the target or a link to it. Thanks for your cooperation.

Email: [SuperCRacker3@hotmail.com](mailto:SuperCRacker3@hotmail.com)

Support Forum: <http://forum.tuts4you.com/index.php?showforum=119>

### **HISTORY**

To view Imports Fixer history in Imports Fixer go: **Help -> History**

### **DISCLAIMER**

Although the greatest care and attentiveness was taken in the development of Imports Fixer ... it may fail in its job. It is not supposed to but in the worst scenario, it might even reboot your computer, so please use it with caution: e.g. be careful too and save your important work regularly ;)

The author (SuperCRacker) or anybody else will **NOT** be responsible for any damage caused by using Imports Fixer. You are free to use it ... or not =)

---