Retargetable Decompiler's IDA Plugin

User Guide

Version 1.0

https://github.com/avast/retdec-idaplugin

https://retdec.com

support@retdec.com

August 18, 2020

Contents

1 Introduction

This document describes the Retargetable Decompiler's plugin for IDA (RetDec plugin). Its goal is to integrate with IDA, give transparent access to the Retargetable Decompiler and provide user-interaction capabilities like navigation or code refactoring. An example of code decompiled by Retdec plugin is shown in Figure 1.

Retargetable Decompiler (RetDec) is a reverse-engineering tool independent of any particular target architecture, file format, operating system, or compiler. It was developed in cooperation of Faculty of Information Technology, Brno University of Technology and AVG Technologies. Since the acquisition of AVG Technologies by Avast in 2016, Avast has continued to develop the decompiler. It is using Capstone disassembler engine and a Capstone2LlvmIR library to translate machine code into a high-level-language representation. Currently, the decompiler supports the x86, x86-64, arm, arm64, mips, and powerpc architectures using the Windows PE, COFF, Unix ELF, macOS Mach-O, Intel HEX, and RAW binary file formats.

RetDec can be used in the following ways:

1. Standalone RetDec: either compiling RetDec repository on your own, or downloading and installing RetDec binary release.

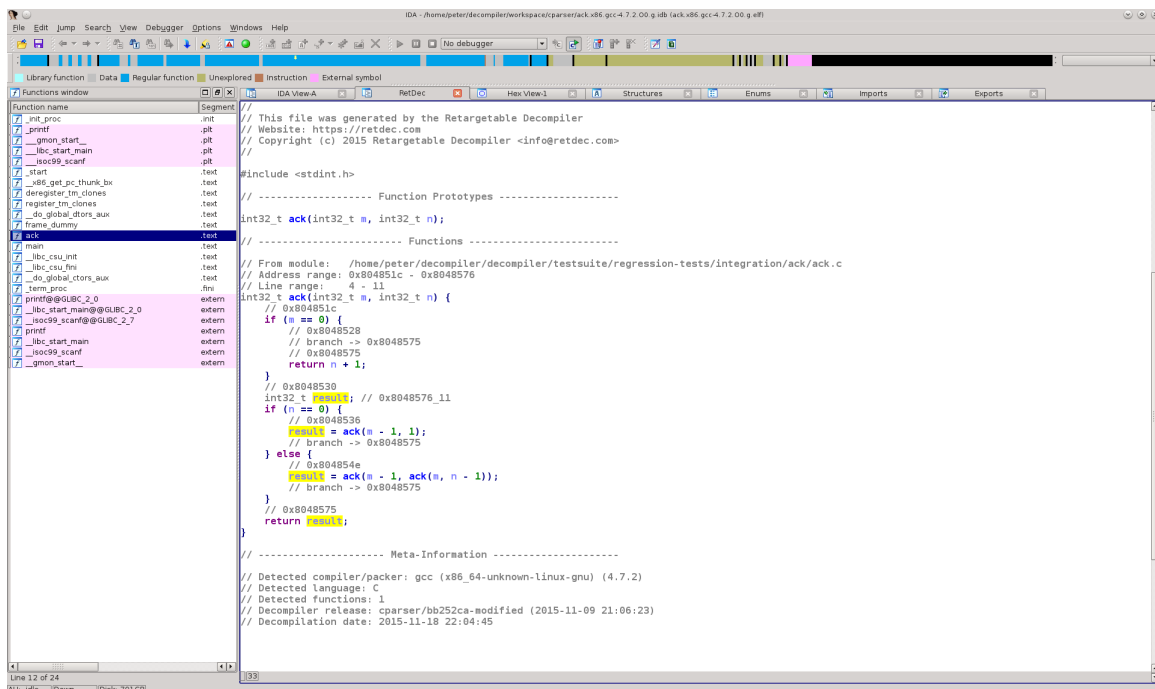2. Standalone RetDec IDA plugin (this guide's topic).

Figure 1: Example of code decompiled by RetDec plugin.

2 Installation

This section describes prerequisites and the installation process of RetDec IDA plugin binary release.

It is also possible to build and install the plugin directly from sources. To do so, follow the Build and Installation instructions instead of this section.

2.1 IDA

The plugin is created using IDA SDK version 7.5. The plugin is compatible with IDA versions 7.5+. The plugin does NOT work with IDA 6.x, IDA 7.0-7.4, or freeware version of IDA 7.0. The plugin comes at both 32-bit and 64-bit address space variants (both are 64-bit binaries). I.e. it works in both `ida` and `ida64`.

2.2 RetDec IDA plugin

The binary release packages are available for Linux and Windows.

2.2.1 Linux

Follow the next steps to install RetDec plugin in a Linux environment:

1. Install 64-bit versions of the following shared-object dependencies:

   ```
   libcrypto.so.1.1 libpthread.so.0 libz.so.1 libdl.so.2 libstdc++.so.6
   libm.so.6 libgcc_s.so.1 libc.so.6
   ```

2. Download the Linux installation package (Table 1) from the project's release page.

3. Extract the package. Directory `retdec-idaplugin` gets created.

4. Copy the contents of the directory, not the directory itself, to the IDA's plugins directory (`<IDA_ROOT>/plugins`).

Table 1: Linux installation package contents.

File	Description
`retdec.so`	64-bit Linux RetDec plugin for 32-bit address space.
`retdec64.so`	64-bit Linux RetDec plugin for 64-bit address space.
`retdec/`	Directory with RetDec resources.
`retdec/user_guide.pdf`	RetDec plugin's user guide (this document).
`retdec/LICENSE`	RetDec IDA plugin's license.
`retdec/LICENSE-THIRD-PARTY`	Licenses of libraries used by RetDec plugin.
`retdec/decompiler-config.json`	RetDec configuration file.
`retdec/*/`	Directories with decompilation resources.

2.2.2 Windows

The Windows version of the plugin requires Windows 7 or later, with the MSVC 2017 runtime[1] installed.

Follow the next steps to install RetDec plugin in a Windows environment:

1. Download the Windows installation package (Table 2) from the [project's release page](#).

2. Extract the package. Directory `retdec-idaplugin` gets created.

3. Copy the contents of the directory, not the directory itself, to the IDA's plugins directory (`<IDA_ROOT>/plugins`).

Table 2: Windows installation package contents.

File	Description
`retdec.dll`	64-bit Windows RetDec plugin for 32-bit address space.
`retdec64.dll`	64-bit Windows RetDec plugin for 64-bit address space.
`retdec/`	Directory with RetDec resources.
`retdec/user_guide.pdf`	RetDec plugin's user guide (this document).
`retdec/LICENSE`	RetDec IDA plugin's license.
`retdec/LICENSE-THIRD-PARTY`	Licenses of libraries used by RetDec plugin.
`retdec/decompiler-config.json`	RetDec configuration file.
`retdec/support/`	Directory with decompilation resources.

3 Configuration

This section describes how to configure RetDec plugin. After you follow these steps, you should have your plugin ready for work.

3.1 IDA's plugin.cfg

The plugin's default mode is set to selective decompilation (see Section 5). It tries to register hotkey `Ctrl+d` for its invocation. If you already use this hotkey for another action or you just want to use a different hotkey, you need to modify IDA's plugin configuration file.

The IDA's plugin configuration file is in `<IDA_ROOT>/plugins/plugins.cfg`. Its format is documented inside the file itself. To configure RetDec plugin to use hotkey `Ctrl+Alt+d` instead of the default `Ctrl+d`, add the following line:

```
; Plugin_name           File_name Hotkey        Arg
; -------------------------------------------------
Retargetable_Decompiler retdec    Ctrl-Alt-d  0
```

---

[1]Visual C++ Redistributable for Visual Studio 2017: https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

4 Plugin Information

This section describes how to find information about RetDec plugin you are currently using and how the plugin communicates information to you.

4.1 About Plugin

Information about RetDec plugin can be found among IDA's information on the registered plugins at `Help/About program` (Figure 2), where you need to click on the `Addons` button (Figure 3). Then, find the `Retargetable Decompiler` entry in the presented list (Figure 4).
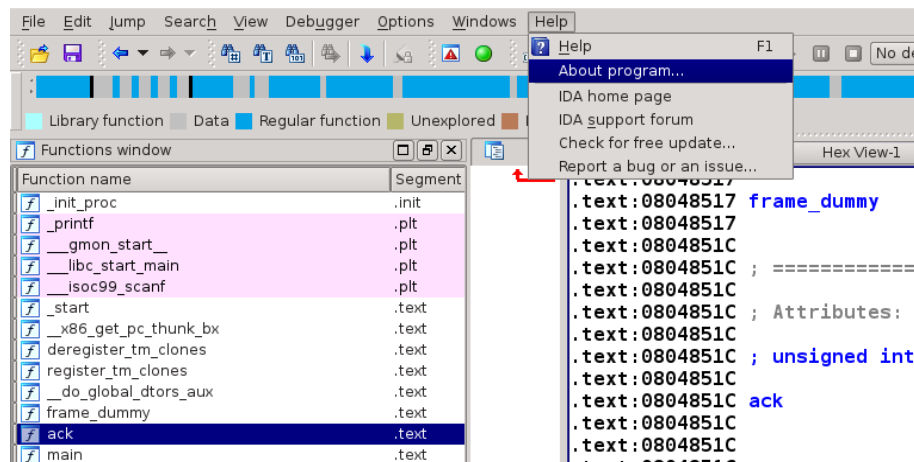

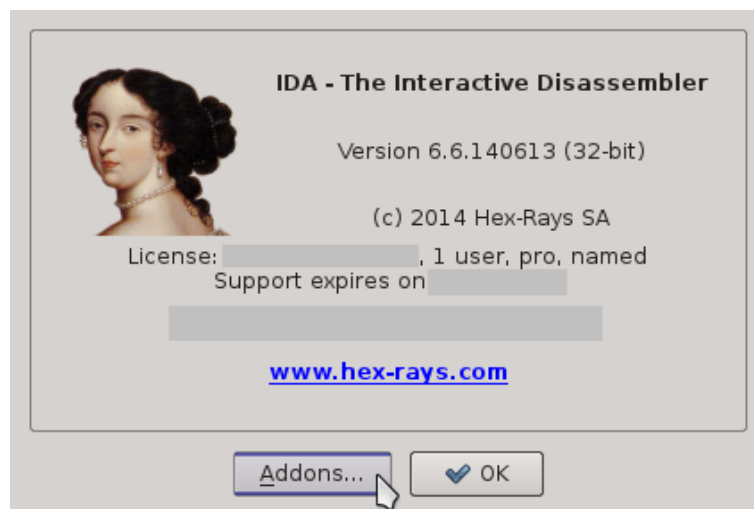
Figure 2: Opening the About IDA dialog from the menu.

Figure 3: Information window about IDA.
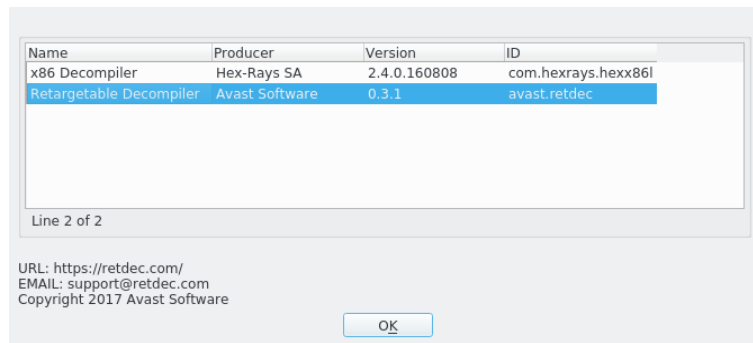
Figure 4: Information window about RetDec plugin.

4.2 Output Window

Right after the start, as well as during the work with RetDec plugin, it communicates with you mainly through the IDA's output window (Figure 5). Here, you are shown several kinds of important messages:

```
[RetDec info]   :    some important piece of information
[RetDec warning]:    something went a little bit wrong
[RetDec error]  :    something went very wrong
```
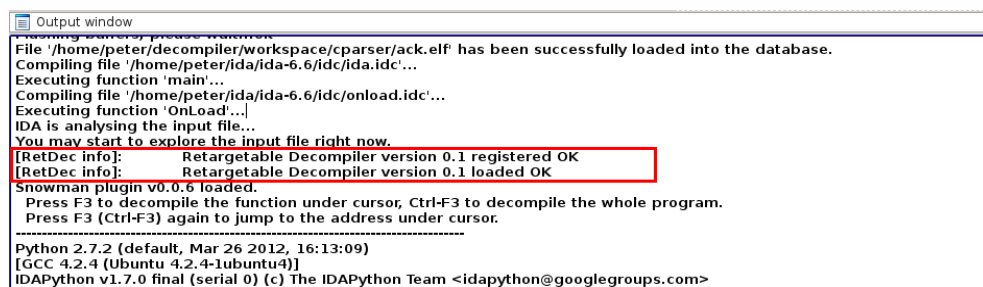

Figure 5: IDA's output window.

4.3 GUI Windows

Sometimes, RetDec plugin wants to be sure you noticed an important message or event. In such a case, it shows you a pop-up window, which forces you to acknowledge it by pressing OK or a similar button.

5 Decompilation

This section describes how to invoke a decompilation. After reading it, you should be able to decompile a selected function, as well as an entire binary that is being analyzed.

5.1 Selective Decompilation

RetDec plugin's primary decompilation mode is selective decompilation. It decompiles the function that is currently under the cursor. It is invoked from the IDA's disassembly window, where you need to bring focus to the desired function and use either the default hotkey `Ctrl+d`, or a hotkey you configured according to Section 3.1. You can also trigger it by running the plugin from the main IDA menu bar (Figure 6).

Once the decompilation is finished, the decompiled source code is displayed in a new IDA viewer window. Here, you can invoke new decompilations by double-clicking on function calls.

At the moment, the RetDec plugin supports only one simultaneous decompilation.

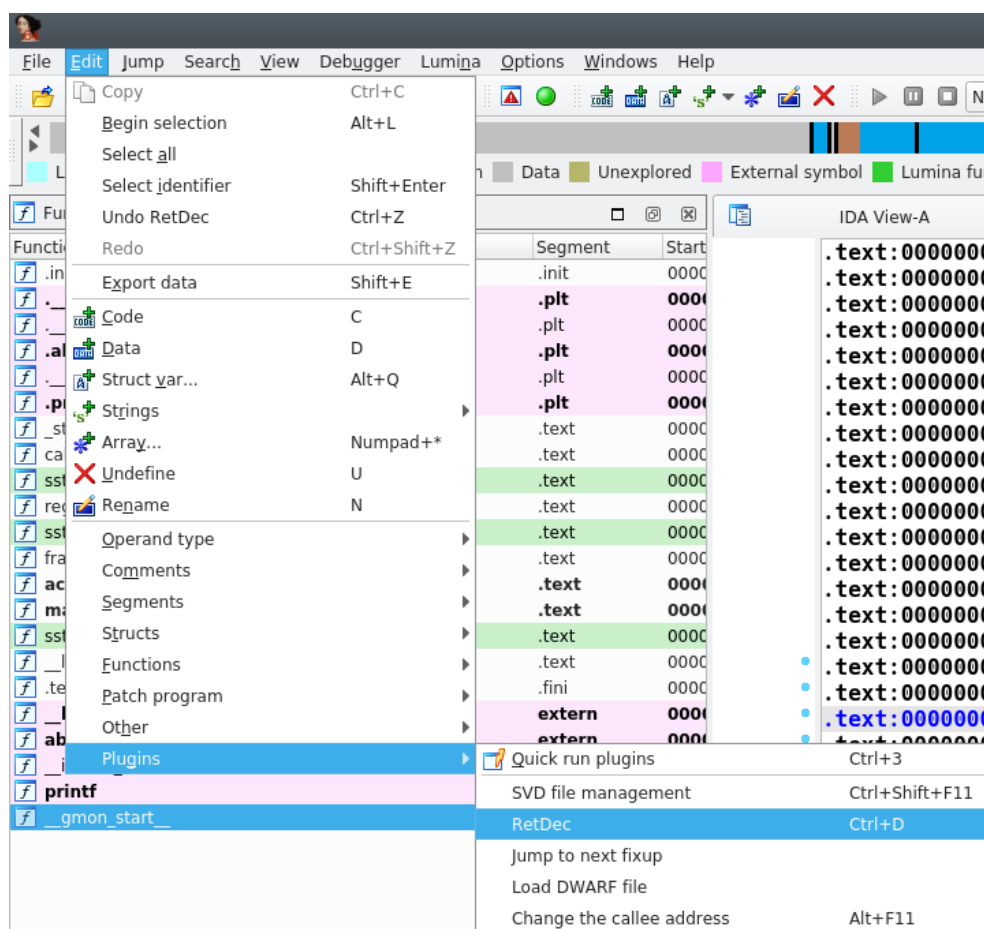

Figure 6: Selective decompilation.

5.2 Full Decompilation

You can also decompile an entire binary that is being analyzed. Either use the hotkey `Ctrl+Shift+d`, or the main IDA menu bar (Figure 7). The result of this decompilation is stored into an output file, whose location is communicated to you through IDA's output window. The result cannot be displayed in the IDA's viewer window.

7

Same as with selective decompilation (Section 5.1), only one decompilation can run at a time.
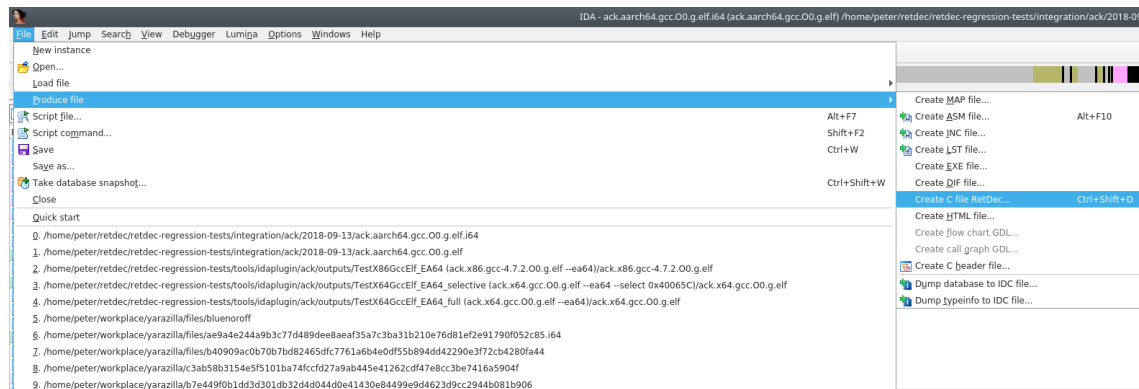

Figure 7: Full decompilation.

6 User Interactions

This section describes various kinds of user interactions that are currently supported by RetDec plugin. As was stated in Section 5, these interactions are applicable only on results from selective decompilations because full decompilations cannot be displayed in IDA's viewer window.

6.1 Basic Interactions

We use the IDA's native custom viewer window to display the decompiled source codes. Therefore, the plugin feels like part of IDA and we get a word occurrences highlighting (Figure 8) out of the box.

6.2 Navigation

RetDec plugin supports function navigation—jumping forward and backward between already decompiled functions, or invoke an entirely new decompilation. When you double-click on a function call, the plugin presents the requested function. If it was already decompiled in the past, the cached result is shown to perform the action faster. You have to either explicitly request a re-decompilation of the previously processed functions, or perform an action that triggers the re-decompilation automatically (see Section 6.3). Re-decompilation can be forced by using the selective decompilation hotkey in IDA's disassembly (re-decompilation of any function), or in the RetDec plugin's viewer window (re-decompilation of currently shown function). If the double-clicked function was not decompiled yet, it is selectively reversed and displayed. In either case, only one function is shown at a time. A navigation entry for the newly presented function is added into a doubly linked navigation list, right after the entry for function from which the invocation

```
int32_t ack(int32_t m, int32_t n) {
    // 0x804851c
    if (m == 0) {
        // 0x8048528
        // branch -> 0x8048575
        // 0x8048575
        return n + 1;
    }
    // 0x8048530
    int32_t result; // 0x8048576_11
    if (n == 0) {
        // 0x8048536
        result = ack(m - 1, 1);
        // branch -> 0x8048575
    } else {
        // 0x804854e
        result = ack(m - 1, ack(m, n - 1));
        // branch -> 0x8048575
    }
    // 0x8048575
    return result;
}
```

Figure 8: Native word occurrence highlighting.

was made. The list is then used for forward and backward navigation between the stored functions. An example of such navigation is depicted in Figure 9.

Initial state:

Decompile fnc_1:   [fnc_1]

Decompile fnc_2:   [fnc_1] ◄─► [fnc_2]

Decompile fnc_3:   [fnc_1] ◄─► [fnc_2] ◄─► [fnc_3]

Move back: [fnc_1] ◄─► [fnc_2] ◄─► [fnc_3]

Move back: [fnc_1] ◄─► [fnc_2] ◄─► [fnc_3]

Decompile fnc_4:   [fnc_1] ◄─► [fnc_4]
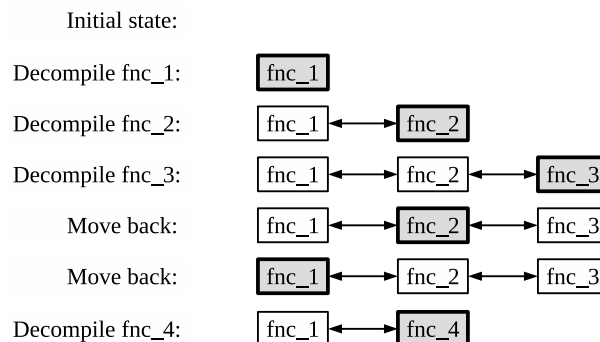
Figure 9: Decompiled function navigation example.

RetDec navigation is integrated with IDA's graphical control elements, so you can use either ⇐ and ⇒ buttons or hotkeys:

- `Esc` to move back.

- `Ctrl+Enter` to move forward.

6.3   Code Refactoring

The RetDec plugin's viewer windows also allows you to refactor displayed source code. We can divide the source-code modifications into two basic categories:

- Those which do not require immediate re-decompilation, like object-identifier re-naming or function-comment insertion.

Figure 10: Context actions available for functions.

Figure 11: Context actions available for global variables.

- Those which automatically trigger re-decompilation of the modified function. These are typically changes that can be used or propagated by reversing analyses. For example, a user-specified object data type can by spread by the data-flow type recovery analysis among other objects.

Refactoring actions are triggered either by hotkeys associated with them, or by pop-up menus shown on right-click. Actions are sensitive to the current context (current word under the cursor). As is shown in Figure 10 and Figure 11, actions available for functions differ from actions for global variables. Available actions at any given position are composed of two sets of actions:

- Actions available for the current context, i.e. for functions, global variables, function calls, etc. This set may be empty.

- Global actions available at all posistions, i.e. navigation, current function comment modification, etc.

The complete catalog of available user actions is listed in Section 7.

7   List of All User Actions

This section provides a complete catalog of available user actions for all possible contexts.

7.1   Function-Declaration/Definition Context

Function actions are available on function declarations or definitions. They are listed in Table 3.

Table 3: Function context user actions.

Action description	Hotkey	Triggers re-decompilation
Jump to IDA's ASM	A	✗
Rename function	N	✗
Change type declaration	Y	✓
Open xrefs window	X	✗
Open calls window	C	✗

7.2 Function-Call Context

Function-call actions are available on function calls. We can divide them into two categories:

- Calls of user defined functions—actions are the same as in function-declaration/definition context (Section 7.1), except the "Change type declaration" action. Also, you can double-click on a call to decompile/display the called function.

- Calls of dynamically linked functions—the only available action is double-click, which takes you on the import stub in the IDA's disassembly view.

7.3 Global-Variable Context

Global-variable actions are available on global-variable definitions and uses. They are listed in Table 4.

Table 4: Global-variable context user actions.

Action description	Hotkey	Triggers re-decompilation
Jump to IDA's ASM	A	✗
Rename global variable	N	✗

7.4 Global Context

Global context actions are available everywhere. They are listed in Table 5.

Table 5: Global context user actions.

Action description	Hotkey	Triggers re-decompilation
Edit current function's comment	;	✗
Move backward (navigation)	ESC	✗
Move forward (navigation)	CTRL+F	✗

8   Support and Feedback

RetDec plugin is still in an experimental beta version. If you have any feedback, suggestions, or bug reports, please open an issue in the GitHub project (preferred), or send them to us either through our website, or through email.

https://github.com/avast/retdec-idaplugin

https://retdec.com

support@retdec.com