



Retargetable Decompiler's IDA Plugin

User Guide

Version 0.3.1

<https://retdec.com>
support@retdec.com
November 27, 2017

Contents

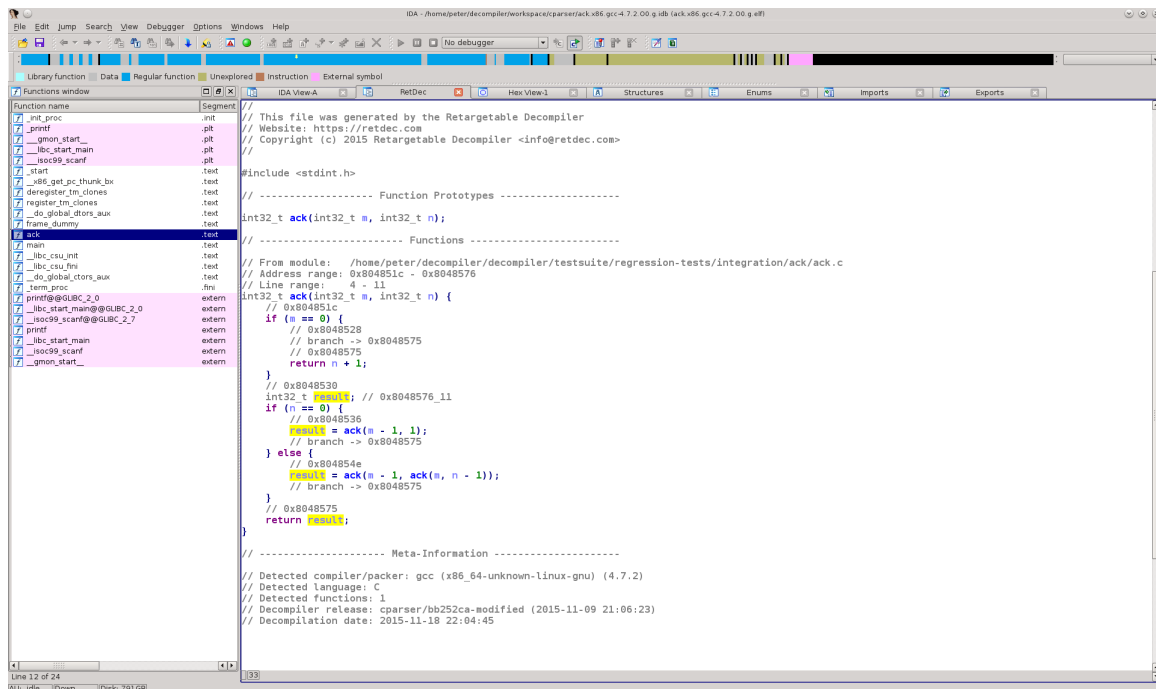
1	Introduction	2
2	Installation	3
2.1	IDA	3
2.2	Linux	3
2.3	Windows	3
2.4	Windows Plugin on Linux	4
3	Configuration	4
3.1	IDA's plugin.cfg	4
3.2	First Run	5
3.3	Configuration from IDA	5
4	Plugin Information	6
4.1	About Plugin	6
4.2	Output Window	7
4.3	GUI Windows	7
5	Decompilation	8
5.1	Selective Decompilation	8
5.2	Full Decompilation	8
6	User Interactions	8
6.1	Basic Interactions	8
6.2	Navigation	9
6.3	Code Refactoring	9
7	List of All User Actions	10
7.1	Function-Declaration/Definition Context	10
7.2	Function-Call Context	10
7.3	Global-Variable Context	11
7.4	Global Context	12
8	Support and Feedback	12

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's decompilation service 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 CZ, s.r.o. It is using architecture description language processor models and extensive pre-processing to translate machine code into a high-level-language representation. Currently, the decompiler supports the MIPS, ARM, x86, PIC32, and PowerPC architectures using the Windows PE, COFF, Unix ELF, Intel HEX, and RAW binary file formats.

The RetDec's decompilation service provides two ways to use the decompiler: 1) web interface and 2) application programming interface (API). RetDec plugin is using the API.



The screenshot shows the RetDec plugin interface within IDA Pro. The left pane displays a list of functions, with 'ack' selected. The main pane shows the decompiled C code for the 'ack' function. The code includes a header for 'stdint.h', a function prototype for 'int32_t ack(int32_t n, int32_t n);', and the function implementation. The implementation starts with a comment indicating the source module and address range. It then defines a local variable 'result' and uses a loop to calculate the result based on the input 'n'. The code is annotated with addresses and branch instructions. The bottom pane shows meta-information about the decompilation process, including the detected compiler/packer, language, functions, and decompilation date.

```
// This file was generated by the Retargetable Decompiler
// Website: https://retdec.com
// Copyright (c) 2015 Retargetable Decompiler <info@retdec.com>

#include <stdint.h>

// ----- Function Prototypes -----
int32_t ack(int32_t n, int32_t n);

// ----- Functions -----

// From module: /home/peter/decompiler/decompiler/testsuite/regression-tests/integration/ack/ack.c
// Address range: 0x804851c - 0x8048576
// Line range: 4 - 11
int32_t ack(int32_t n, int32_t n) {
    // 0x804851c
    if (n == 0) {
        // 0x8048528
        // branch -> 0x8048575
        return n + 1;
    }
    // 0x8048530
    int32_t result; // 0x8048576_11
    if (n == 0) {
        // 0x8048536
        result = ack(n - 1, 1);
        // branch -> 0x8048575
    } else {
        // 0x804854e
        result = ack(n - 1, ack(n, n - 1));
        // branch -> 0x8048575
    }
    // 0x8048575
    return result;
}

// ----- Meta-Information -----
// Detected compiler/packer: gcc (x86_64-unknown-linux-gnu) (4.7.2)
// Detected language: C
// Detected functions: 1
// Decompiler release: cparser/bb252ca-modified (2015-11-09 21:06:23)
// Decompilation date: 2015-11-18 22:04:45
```

Figure 1: Example of code decompiled by RetDec plugin.

2 Installation

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

2.1 IDA

The plugin is created using IDA SDK version 6.6. Therefore, you need IDA version 6.6 or later to run it.

2.2 Linux

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

1. Install 32-bit versions of the following shared-object dependencies:

```
libc.so.6 libgcc_s.so.1 libm.so.6 libpthread.so.0 libstdc++.so.6
```

2. Download the Linux installation package (Figure 1) from our website.
3. Copy `retdec.plx` to the IDA's plugin directory (`<IDA_root>/plugins`).

Table 1: Linux installation package contents.

File	Description
license	Directory with licenses.
license/3rdparty.txt	Licenses of libraries used by RetDec plugin.
license/license_us.htm	RetDec plugin's end user agreement (EULA).
user_guide.pdf	RetDec plugin's user guide (this document).
retdec.plx	32-bit Linux RetDec plugin.

2.3 Windows

The Windows version of the plugin requires Windows 7 or later, with the MSVC 2015 runtime¹ installed.

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

1. Download the Windows installation package (Figure 2) from our website.
2. Copy `retdec.plw` to the IDA's plugin directory (`<IDA_root>/plugins`).

¹Visual C++ Redistributable for Visual Studio 2015: <https://www.microsoft.com/en-us/download/details.aspx?id=48145>

Table 2: Windows installation package contents.

File	Description
license	Directory with licenses.
license/3rdparty.txt	Licenses of libraries used by RetDec plugin.
license/license_us.htm	RetDec plugin's end user agreement (EULA).
user_guide.pdf	RetDec plugin's user guide (this document).
retdec.plw	32-bit Windows RetDec plugin.

2.4 Windows Plugin on Linux

It is also possible to run the Windows version of IDA with the Windows version of RetDec plugin on Linux using Wine². Install RetDec plugin as described in Section 2.3 and if it does not run out of the box, try a workaround³.

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. Moreover, the plugin supports one more decompilation mode and a hotkey invocation for the plugin's configuration. If you want to use any of them, you also have to modify the config 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, add the following lines at the beginning⁴ of the file:

```
; Plugin_name          File_name Hotkey      Arg
; -----
Retargetable_Decompiler retdec    Ctrl-d      0
Retargetable_Decompiler retdec    Ctrl-Shift-d 1
Retargetable_Decompiler retdec    Ctrl-Shift-c 2
```

These lines tell IDA which hotkeys invoke the plugin and what argument is passed to it. The plugin's behavior after invocation is determined by the passed argument. Possible argument values are summarized in Table 3. In the provided example, we mapped selective

²<https://www.winehq.org/>

³https://bugs.winehq.org/show_bug.cgi?id=39437#c6

⁴Newer versions of IDA behave strangely when the lines are appended at the end, so just put them at the start.

decompilation to hotkey CTRL+D (plugin's default), full decompilation to CTRL+SHIFT+D, and plugin configuration to CTRL+SHIFT+C. However, you may choose whichever hotkeys you like, provided they do not clash with other plugins or IDA.

Table 3: Description of RetDec plugin's invocation arguments.

Argument value	Description
0	Invokes selective decompilation. See Section 5.
1	Invokes full decompilation. See Section 5.
2	Invokes plugin configuration inside IDA. See Section 3.3.

3.2 First Run

When you run IDA for the first time after you installed RetDec plugin, you are asked to configure the plugin. The configuration dialog is shown in Figure 2. To use the plugin, you have to provide your API key⁵ and check that you agree with our terms of use. If you choose not to agree, you will not be able to run any decompilations. No matter what you choose, you will not be presented with the dialog next time you run IDA. If you want to change the settings later, you need to manually invoke the plugin's configuration from IDA (see Section 3.3).

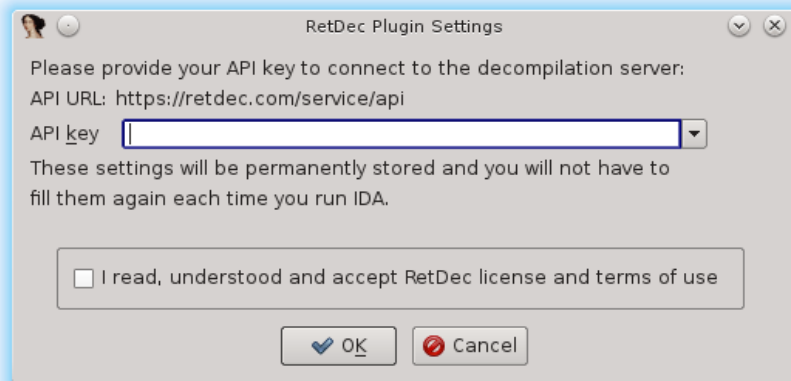


Figure 2: RetDec plugin's configuration dialog.

3.3 Configuration from IDA

The same dialog (Figure 2) that is displayed at the first run of IDA after plugin installation can be opened from IDA anytime later. There are the following two ways to do it:

- If you configured a hotkey for the plugin's configuration argument value according to Section 3.1, you can use it to invoke the configuration dialog.

⁵To access it, register to <https://retdec.com>, login, and click on Account.

- You can also open the configuration dialog from the Options/RetDec plugin options menu (Figure 3).

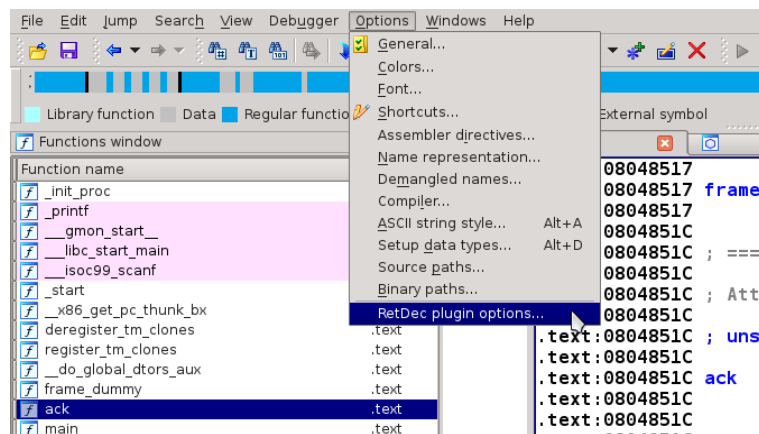


Figure 3: Opening the plugin's configuration dialog from the menu.

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 4), where you need to click on the Addons button (Figure 5). Then, find the Retargetable Decompiler entry in the presented list (Figure 6).

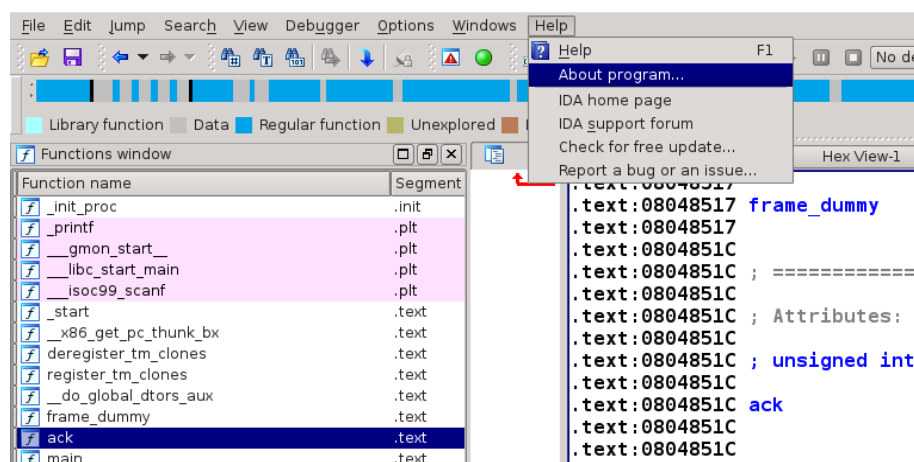


Figure 4: Opening the About IDA dialog from the menu.

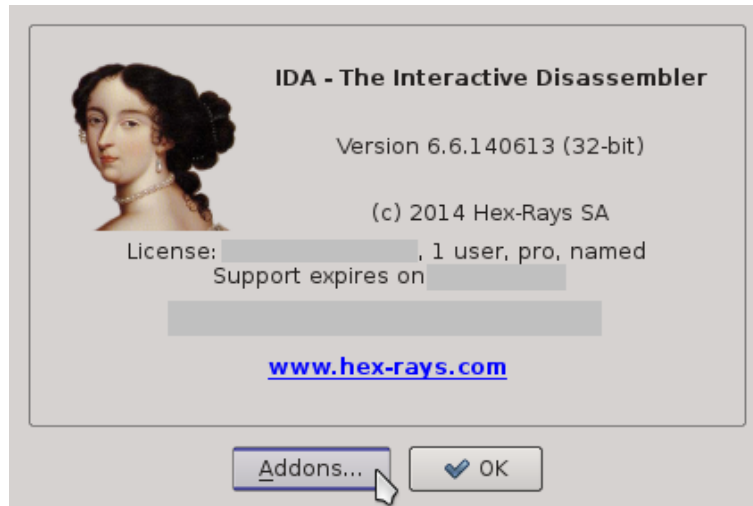


Figure 5: Information window about IDA.

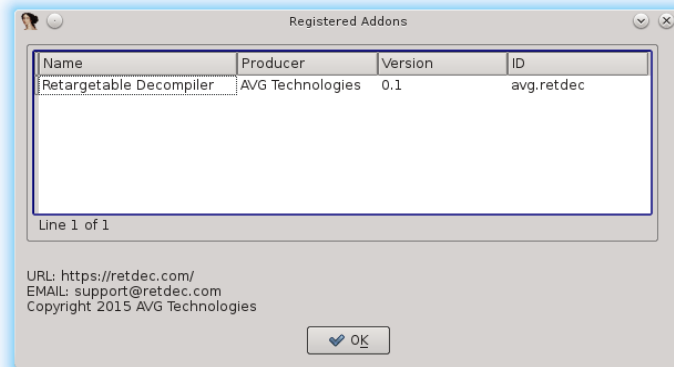


Figure 6: Information window about RetDec plugin.

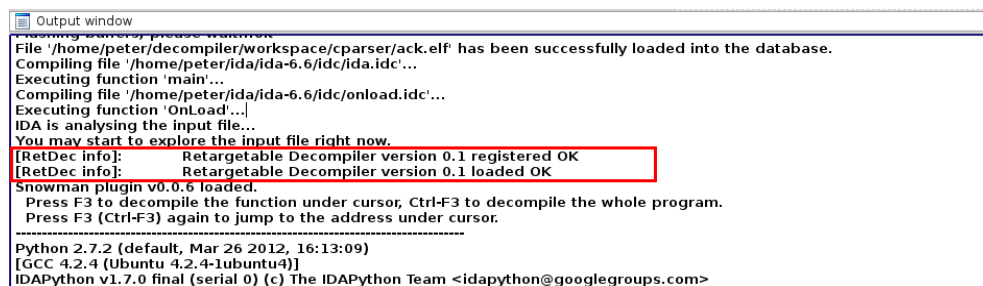
4.2 Output Window

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 7). 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
```

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.



```
Output window
Loading kernel, please wait...
File '/home/peter/decompiler/workspace/cparser/jack.elf' has been successfully loaded into the database.
Compiling file '/home/peter/ida/ida-6.6/idc/ida.idc'...
Executing function 'main'...
Compiling file '/home/peter/ida/ida-6.6/idc/onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
[RetDec info]: Retargetable Decompiler version 0.1 registered OK
[RetDec info]: Retargetable Decompiler version 0.1 loaded OK
Snowman plugin v0.0.6 loaded.
Press F3 to decompile the function under cursor, Ctrl-F3 to decompile the whole program.
Press F3 (Ctrl-F3) again to jump to the address under cursor.
-----
Python 2.7.2 (default, Mar 26 2012, 16:13:09)
[GCC 4.2.4 (Ubuntu 4.2.4-1ubuntu4)]
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
```

Figure 7: IDA's output window.

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.

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.

5.2 Full Decompilation

If you configured a hotkey for full decompilation in Section 3.1, you can use it to decompile an entire binary that is being analyzed. 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.

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.

```

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.

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 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.

Unfortunately, we were not able to integrate navigation with IDA’s graphical control elements, so it can be done only through keyboard hotkeys:

- ESC to move back.
- CTRL+F 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:

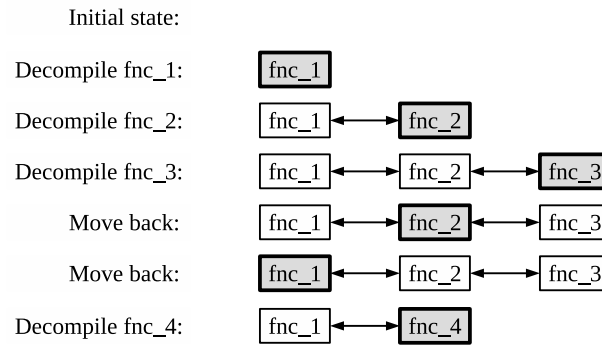


Figure 9: Decompiled function navigation example.

- Those which do not require immediate re-decompilation, like object-identifier re-naming or function-comment insertion.
- 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 be 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 positions, 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 4.

7.2 Function-Call Context

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

```

// From module: /home/peter/decompiler/decompiler.
// Address range: 0x804851c - 0x8048576
// Line range: 4 - 11
int32_t ack(int32_t m, int32_t n) {
// 0x804851c
if (m)
// 0x804851e
// 0x804851f
// 0x8048520
// 0x8048521
// 0x8048522
// 0x8048523
// 0x8048524
// 0x8048525
// 0x8048526
// 0x8048527
// 0x8048528
// 0x8048529
// 0x804852a
// 0x804852b
// 0x804852c
// 0x804852d
// 0x804852e
// 0x804852f
// 0x8048530
// 0x8048531
// 0x8048532
// 0x8048533
// 0x8048534
// 0x8048535
// 0x8048536
// 0x8048537
// 0x8048538
// 0x8048539
// 0x804853a
// 0x804853b
// 0x804853c
// 0x804853d
// 0x804853e
// 0x804853f
// 0x8048540
// 0x8048541
// 0x8048542
// 0x8048543
// 0x8048544
// 0x8048545
// 0x8048546
// 0x8048547
// 0x8048548
// 0x8048549
// 0x804854a
// 0x804854b
// 0x804854c
// 0x804854d
// 0x804854e
// 0x804854f
// 0x8048550
// 0x8048551
// 0x8048552
// 0x8048553
// 0x8048554
// 0x8048555
// 0x8048556
// 0x8048557
// 0x8048558
// 0x8048559
// 0x804855a
// 0x804855b
// 0x804855c
// 0x804855d
// 0x804855e
// 0x804855f
// 0x8048560
// 0x8048561
// 0x8048562
// 0x8048563
// 0x8048564
// 0x8048565
// 0x8048566
// 0x8048567
// 0x8048568
// 0x8048569
// 0x804856a
// 0x804856b
// 0x804856c
// 0x804856d
// 0x804856e
// 0x804856f
// 0x8048570
// 0x8048571
// 0x8048572
// 0x8048573
// 0x8048574
// 0x8048575
// 0x8048576
return result;
}

```

Figure 10: Context actions available for functions.

```

// ----- Global Variables -----
int32_t CTOR_LIST = -1; // 0x80497f4

// ----- Functions -----
// Address range: 0x8048680 - 0x80486a9
int32_t do_global_ctors_aux(void) {
// 0x8048680
if (CTOR_LIST == -1) {
// 0x80486a4
return -1;
}
int32_t v1 = 0x80486a4;
unknown_ffffffff(v1);
// branch -> 0x8048698
while (*(int32_t *)v1 != -1) {
// 0x8048698
v1 -= 4;
unknown_ffffffff(v1);
// continue -> 0x8048698
}
// 0x80486a4
return -1;
}

```

Figure 11: Context actions available for global variables.

Table 4: Function context user actions.

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

- 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 5.

Table 5: Global-variable context user actions.

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

7.4 Global Context

Global context actions are available everywhere. They are listed in Table 6.

Table 6: Global context user actions.

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

8 Support and Feedback

RetDec plugin is still in an experimental beta version. If you have any feedback, suggestions, or bug reports, please send them to us, either through our website or through email:

<https://retdec.com>

support@retdec.com