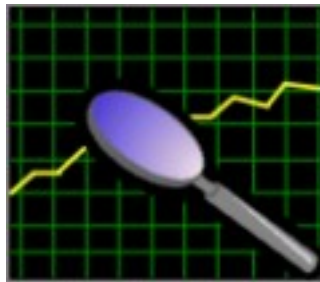


# ***FindLeak – Manual***

FindLeak Version 5.0  
April 2008



[www.findleak.de](http://www.findleak.de)

**Contents:**

<a href="#">1. Getting Information about FindLeak</a>	<a href="#">3</a>
<a href="#">2. FindLeak Demo</a>	<a href="#">3</a>
<a href="#">3. Welcome to FindLeak</a>	<a href="#">4</a>
<a href="#">4. Memory Leaks</a>	<a href="#">5</a>
<a href="#">5. New Features of FindLeak 5.0</a>	<a href="#">6</a>
<a href="#">6. Workshop: Getting to know FindLeak</a>	<a href="#">9</a>
<a href="#">7. A tour through FindLeak</a>	<a href="#">16</a>
<a href="#">8. Working with FindLeak</a>	<a href="#">18</a>
<a href="#">9. Registering FindLeak</a>	<a href="#">21</a>
<a href="#">10. Problems und Solutions</a>	<a href="#">22</a>
<a href="#">11. Frequently asked questions</a>	<a href="#">22</a>
<a href="#">12. Menu Reference</a>	<a href="#">27</a>

# 1. Getting Information about FindLeak

This document introduces the features of FindLeak, shows how to for search memory leaks, and answers questions for common problems. However, if you have further questions, feel free to contact us at one of the addresses printed below.

Visit our FindLeak homepage: [www.findleak.de](http://www.findleak.de)

For technical questions please contact our support team: [support@findleak.de](mailto:support@findleak.de)

For licensing related information please contact us at: [sales@findleak.de](mailto:sales@findleak.de)

By phone: **+49 (7244) 73 41 11**

FindLeak licenses can be order by mail:

**cjt Systemsoftware AG**  
**Am See 14a**  
**76297 Stutensee**  
**Germany**

or by fax:

**+ 49 (7244) 73 41 20**

Please refer to chapter 9 “Registering FindLeak” for information on how to acquire a FindLeak license.

## 2. FindLeak Demo

The FindLeak demo, as available on the FindLeak homepage, helps you learn more about FindLeak's features. It is restricted in such that you can only use it to debug the Microsoft Visual C++ example application, as provided with the demo's download. After purchasing a license key you can turn the FindLeak demo into the unlimited version.

### **3. Welcome to FindLeak**

FindLeak is a Windows development tool for searching memory leaks. It has been successfully used in several large scale system software development projects. Many years of the know-how and experience of error tracing and debugging under different Win32 platforms went into the development of FindLeak. It can be used under Windows XP (with SP2) and Windows Vista (32 Bit).

It is not possible to use FindLeak in conjunction with other memory debugging tools, as FindLeak hooks itself in-between the monitored process and the operating system. Furthermore, monitoring an application with FindLeak increases the application's memory requirements. Measuring the memory consumption of an application attached in FindLeak thus leads to inaccurate results and should best be avoided.

There are several types of memory leaks, FindLeak is unable to detect. As such, an application might still have memory leaks, even though FindLeak did not detect any leaks.

## 4. Memory Leaks

### Memory Leaks

Memory leaks are errors that happen during software development when a memory resource gets allocated but never freed up. With memory leaks, the application's amount of allocated but unused memory rises over time. This can have a negative impact on the overall system, ranging from performance issues to crashes.

### Multiple Memory De-Allocations

If BSTRs and safearrays were allocated using the Oleaut32.dll, freeing these resources more than once leads to problems. The methods that are freeing these resources do not report an error in case of freeing a resource more than once. This is shown in the following scenario:

#### Case A:

1. Allocating of a BSTR at address 0x00140000 within thread A
2. Freeing of this BSTR at address 0x00140000 within thread A
3. Again freeing of this BSTR at address 0x00140000 with thread A (this does not bring up an error)

#### Case B:

1. Allocating of a BSTR at address 0x00140000 within thread A
2. Freeing of this BSTR at address 0x00140000 within thread A
3. Allocating of a new BSTR at address 0x00140000 within thread B
4. Again freeing of this BSTR at address 0x00140000 within thread A (B is now using memory that has already been freed. Accessing this memory can lead to a crash)
5. Freeing of this BSTR at address 0x00140000 within thread B (this does not bring up an error)

Because case B3 is seldom this might lead to sporadic crashes that can only be found by a code review.

## 5. New Features of FindLeak 5.0

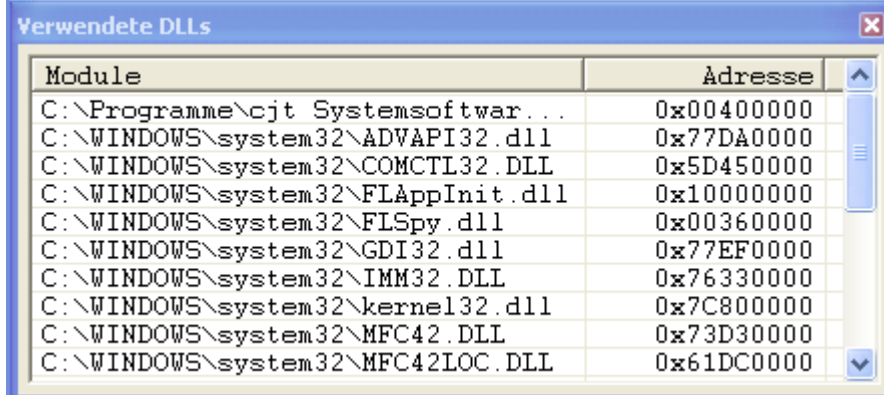
FindLeak 5.0 introduces several new features:

### ***Pause and continue monitoring of a process:***

Monitoring of a process can be stopped with the "Detach" menu item of the process context menu. The "Attach" menu item re-starts monitoring of the process.

### ***Show used DLLs***

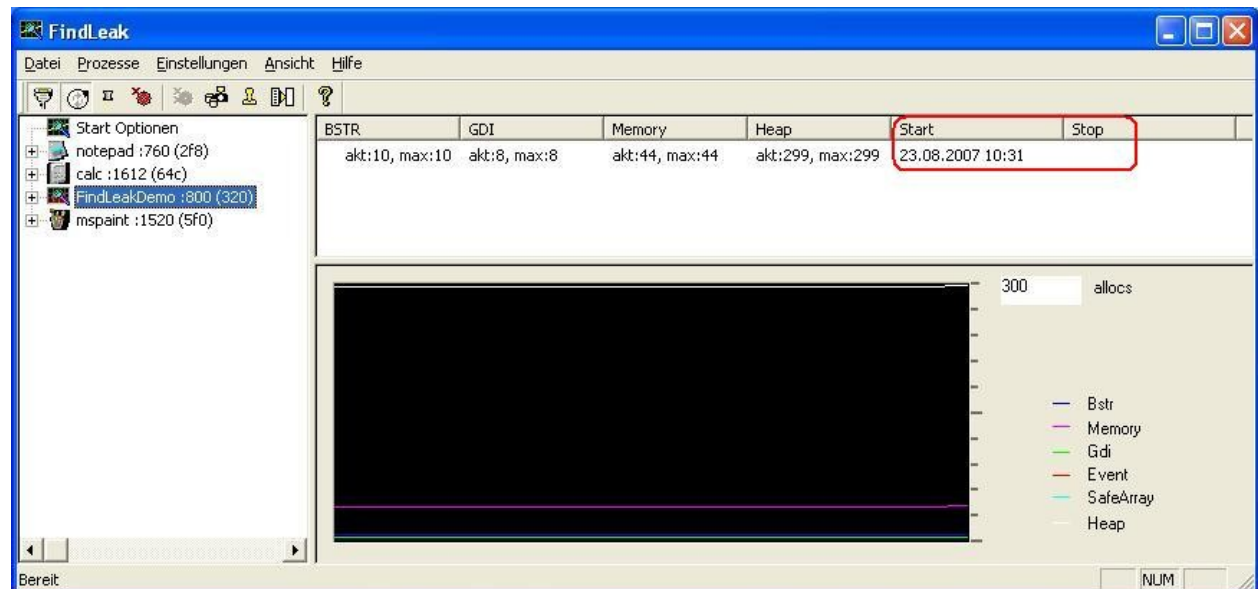
All DLLs as used by a process can be listed via the "Modules" entry of the process context menu.



Module	Adresse
C:\Programme\cjt Systemsoftwar...	0x00400000
C:\WINDOWS\system32\ADVAPI32.dll	0x77DA0000
C:\WINDOWS\system32\COMCTL32.DLL	0x5D450000
C:\WINDOWS\system32\FLAppInit.dll	0x10000000
C:\WINDOWS\system32\FLSpy.dll	0x00360000
C:\WINDOWS\system32\GDI32.dll	0x77EF0000
C:\WINDOWS\system32\IMM32.DLL	0x76330000
C:\WINDOWS\system32\kernel32.dll	0x7C800000
C:\WINDOWS\system32\MFC42.DLL	0x73D30000
C:\WINDOWS\system32\MFC42LOC.DLL	0x61DC0000

### ***Record when a process started and stopped***

FindLeak records when a process starts. In addition, it records the timestamp of when a process stopped.



### ***Cyclic marking of new allocations***

The "Settings" – "Cyclic mark..." menu allows to set the time span FindLeak will wait in seconds, until it marks all allocations as new.

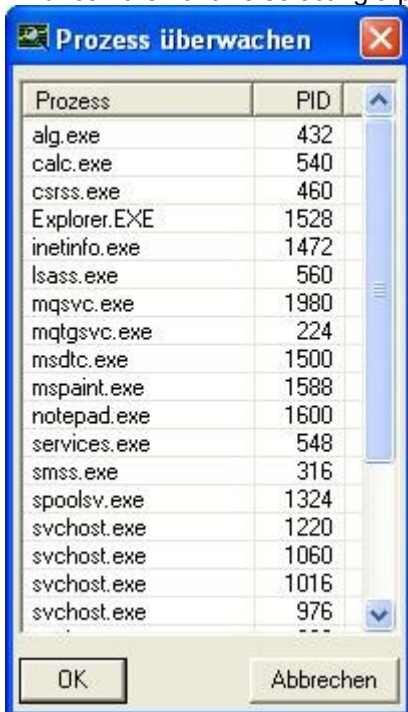


### ***Detach from a running process***

It is now possible to detach FindLeak from a running process by closing FindLeak via the File – Exit menu, or via "Detach" in the process context menu.

### ***Attach to a running process***

It is possible to attach FindLeak to a running process via the "Process" – "Attach to process..." menu. FindLeak then allows selecting a process to be attached to from the list of currently running processes:



Please note that FindLeak will not monitor memory allocations that happened before FindLeak got attached to a process. In other words, allocations or double freeing errors from before attaching will not be listed or lead to a break.

### ***Improved call stack display***

In addition to the addresses of the stack, FindLeak now shows the corresponding modules. If a PDB file is available for the process, FindLeak will also show the matching file, function, and line number.

Double or right clicking on a line in the call stack will open the source code in MS Visual Studio. If MS Visual Studio is not installed, the source code will be shown in Notepad instead.

It is furthermore possible to copy an address via drag&drop into an editor or MS Visual Studio.

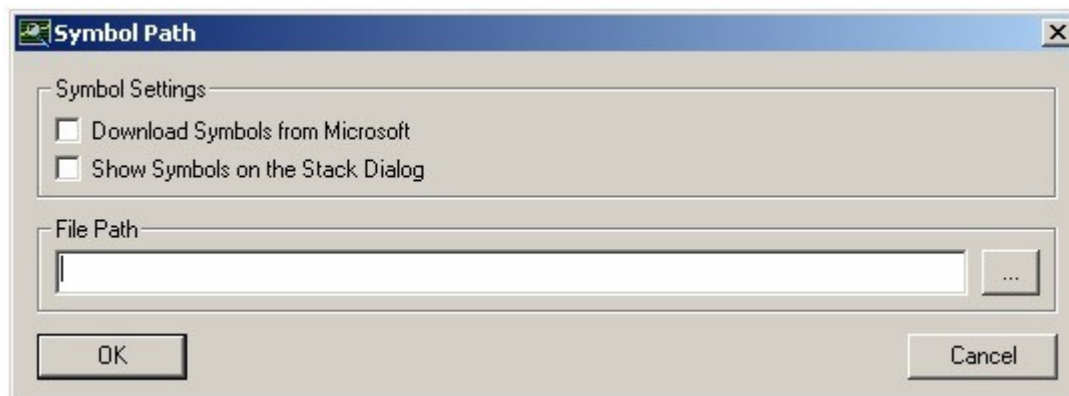


The screenshot shows a window titled "Stack: Heap : 137" with a table of stack entries. The table has five columns: Adresse, Module, Funktion, Datei, and Zeile. The entries are as follows:

Adresse	Module	Funktion	Datei	Zeile
774CDE7B	ole32.dll	StringFromGUID2		
774D2A46	ole32.dll	ColInitialize		
004012B6	FindLeakDe...	CFindLeakDemoApp::InitATL	FindLeakDemo....	145
00401102	FindLeakDe...	CFindLeakDemoApp::InitInstance	FindLeakDemo....	46
73D3CF74	MFC42.DLL	Ordinal1576		
7C816D4F	kernel32.dll	RegisterWaitForInputIdle		

### ***Set the symbol path***

The settings – symbol path menu opens the "Symbol Path..." dialog.



If the "Download Symbols from Microsoft" box is checked, FindLeak will try to load symbols from the Microsoft symbol server. If set, a local path for the symbol cache has to be set in the "File Path" box.

The "Show symbols on the Stack Dialog" box enables the display of function, file and line number of a call stack entry (see above).

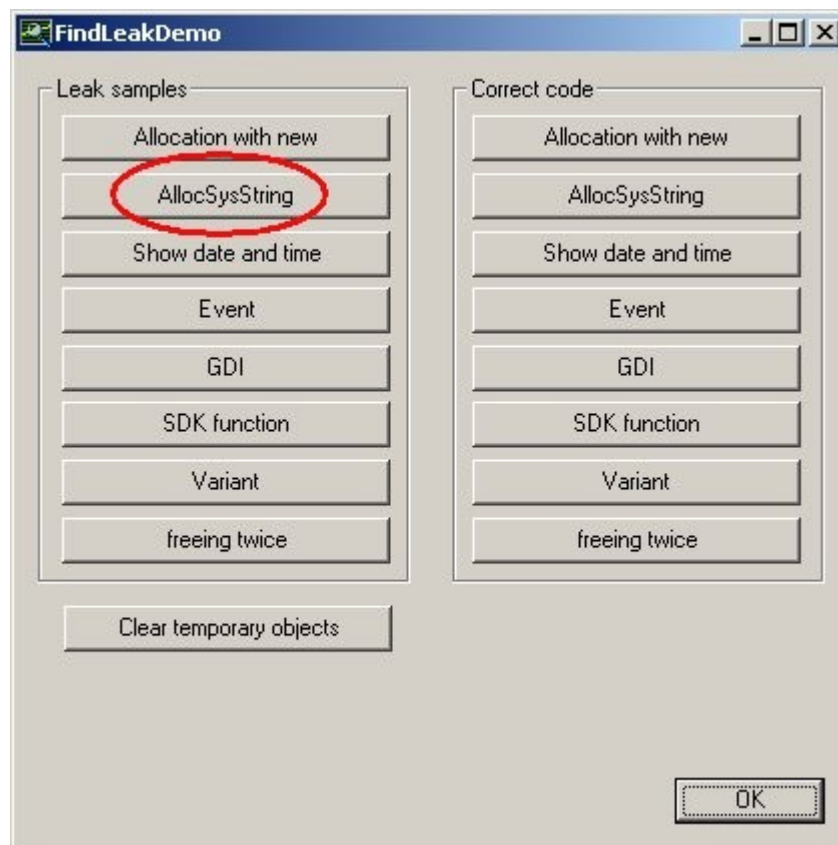


## 6. Workshop: Getting to know FindLeak

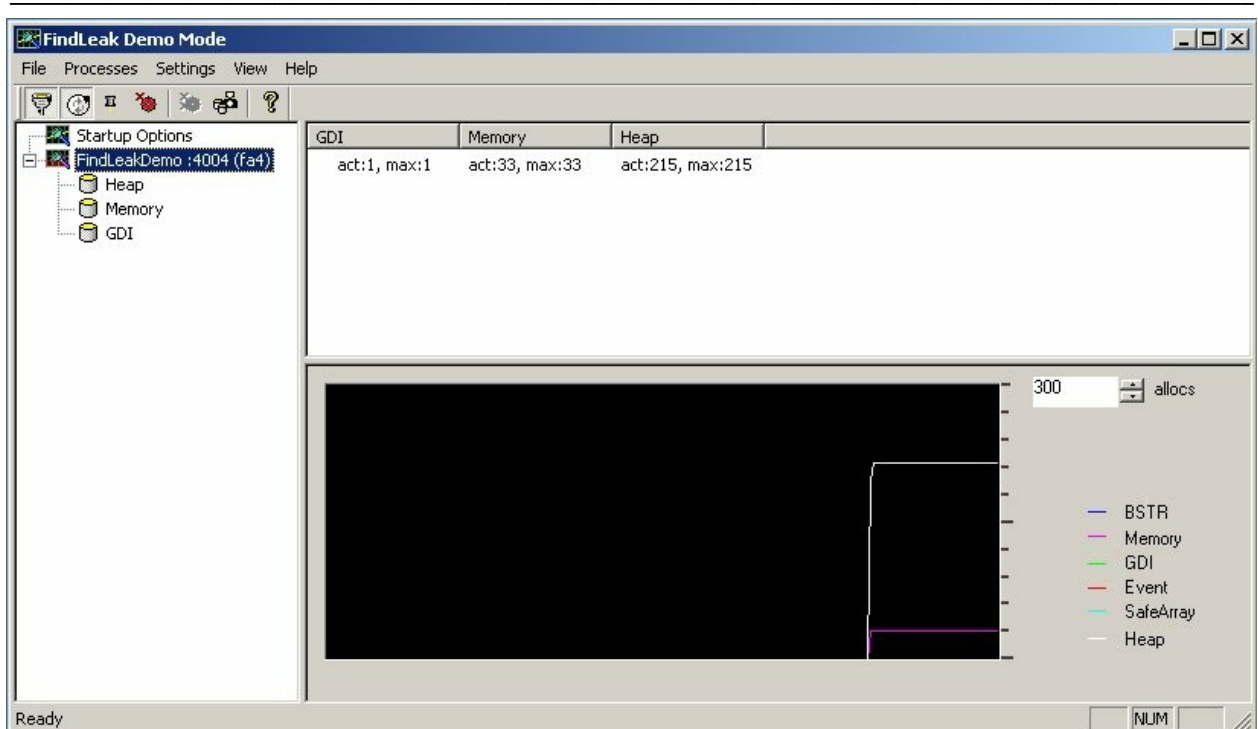
The following workshop shows you how to find a memory leak with FindLeak. To be more specific, it shows a missing memory de-allocation after calling `AllocSysString()`. Though it is only a simple memory leak example, it already gives you an idea on how the powerful features of FindLeak can help you to track down memory related problems.

Start FindLeak.

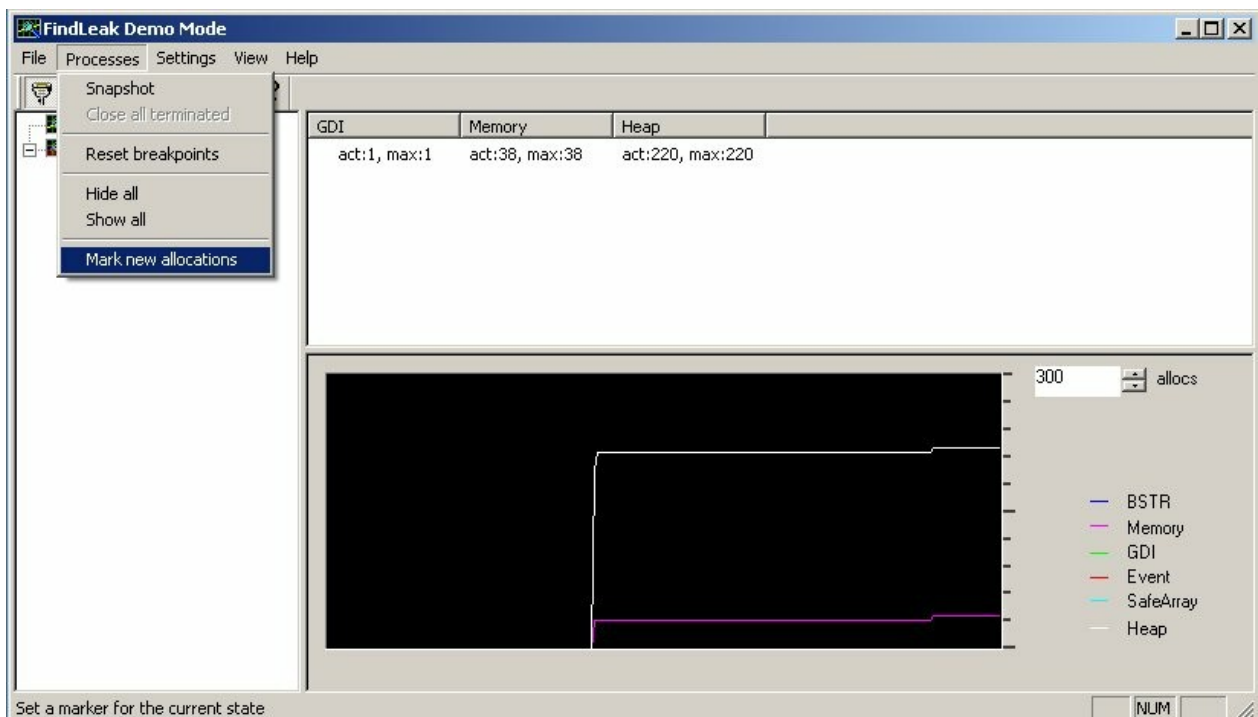
Start the application to be examined, FindLeakDemo in this case. (it is included with it's source code in the FindLeak evaluation download.) Within this sample application click the button "AllocSysString" (on the left side in the category "Leak samples"). This will then execute code for allocating a string without releasing the allocated memory.



FindLeak's process view should already list the FindLeakDemo application. After expanding FindLeak's tree view on the left side (by clicking on the "+" symbol in FindLeak's process view) the currently allocated memory resources for heap, memory, and GDI objects can be seen in the list view on the right hand side.

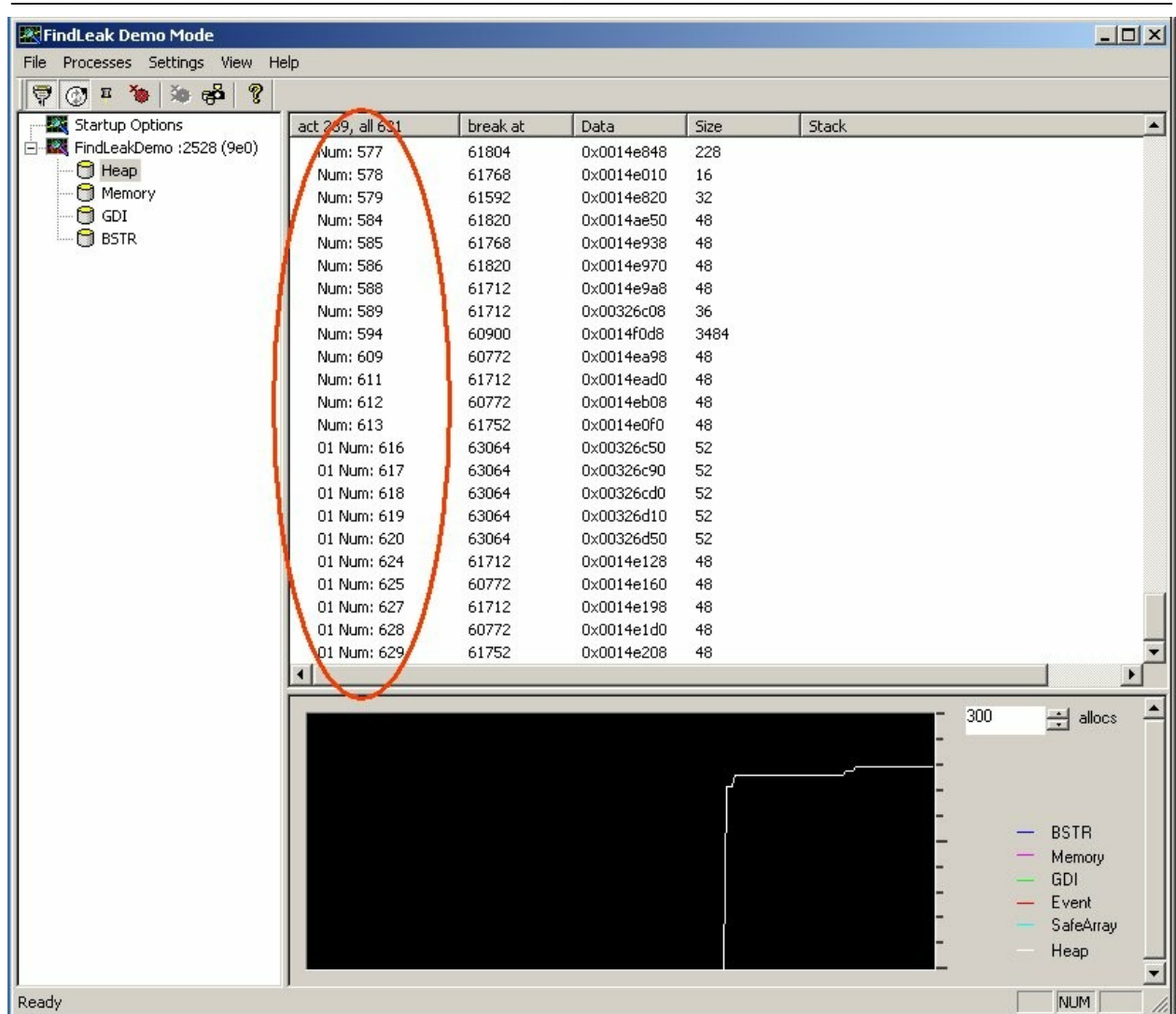


Right-click on the "FindLeakDemo" entry to pop up the context menu. Select "Mark new allocations" in order to visualize further memory allocations.

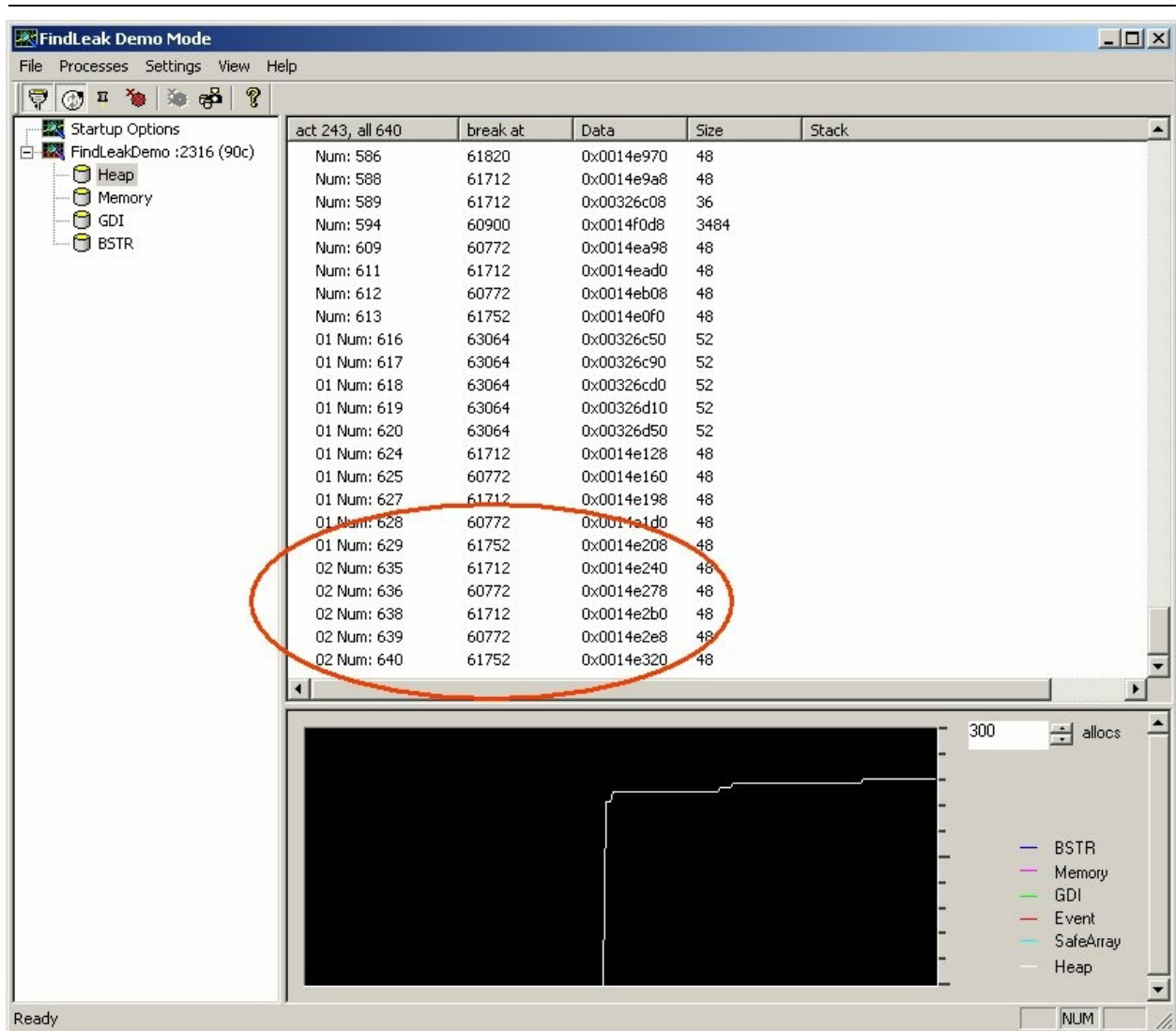


Re-execute the incorrect code by clicking the FindLeakDemo's "AllocSysString" button for another time.

Switch back to FindLeak's process view and expand the "FindLeakDemo" entry. Select "Heap" to see newly added entries in the list view on the right hand side, beginning with the prefix "01". These entries represent memory locations that were allocated during the second "AllocSysString" pass.

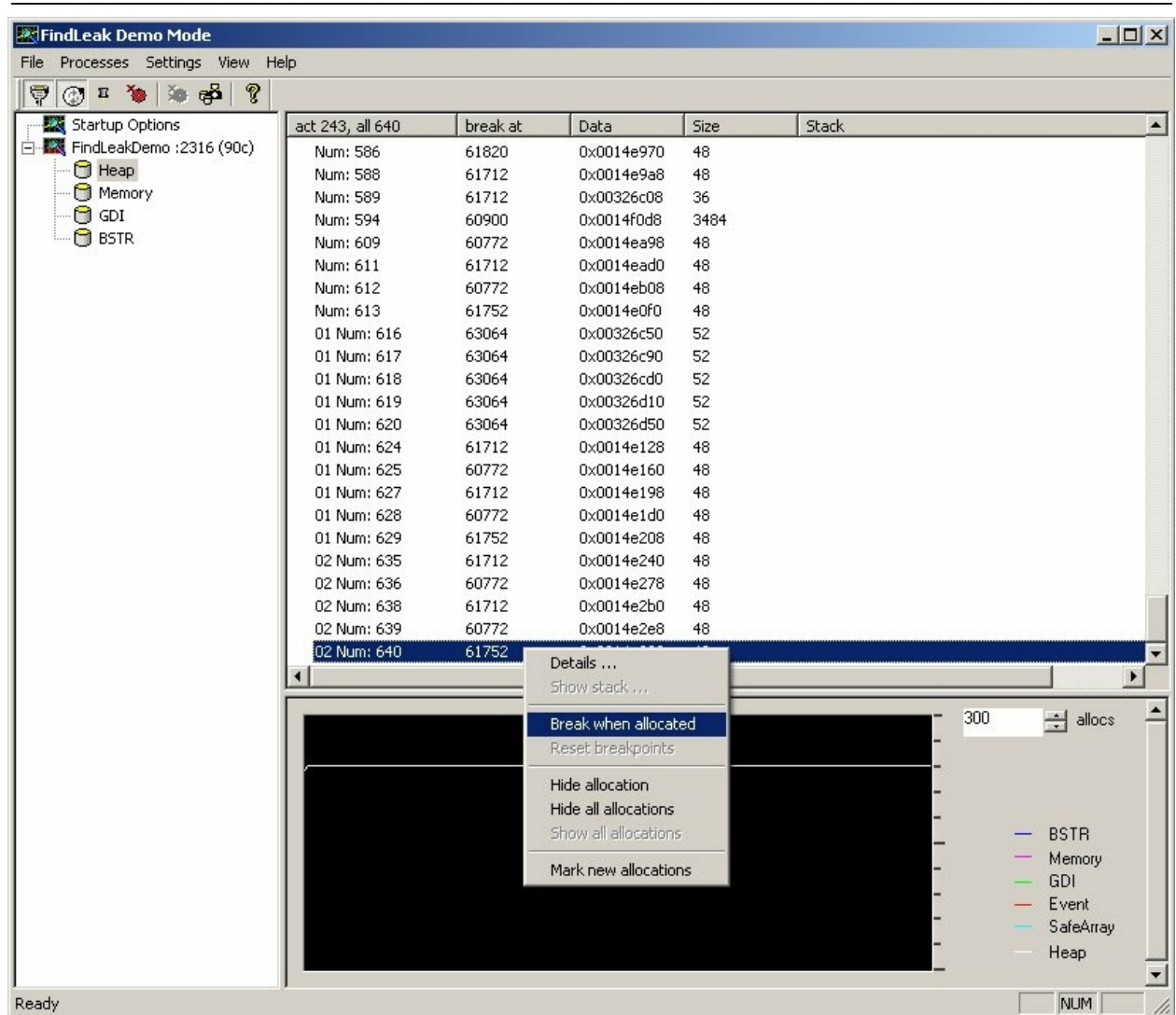


Re-iterate the last three steps in order to demonstrate further memory allocations. With each pass, new entries appear in the list view, marked with the prefix "02", "03", and so on.

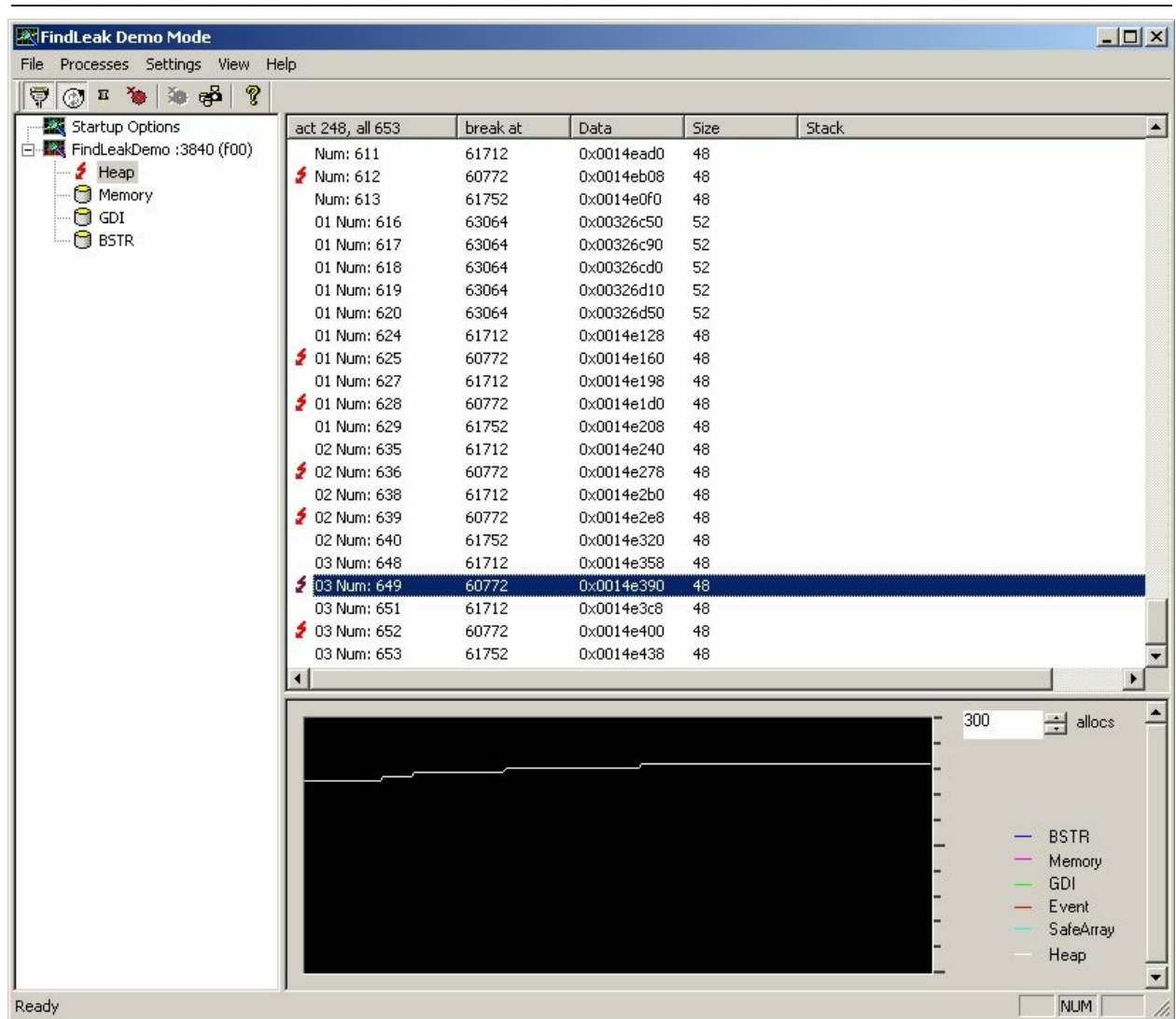


With each pass you can see a certain quantity of memory being allocated, but never being released. Furthermore, the corresponding value in the "break point" column remains the same over several iterations. This indicates that the memory allocation happens in a particular part of the source code.

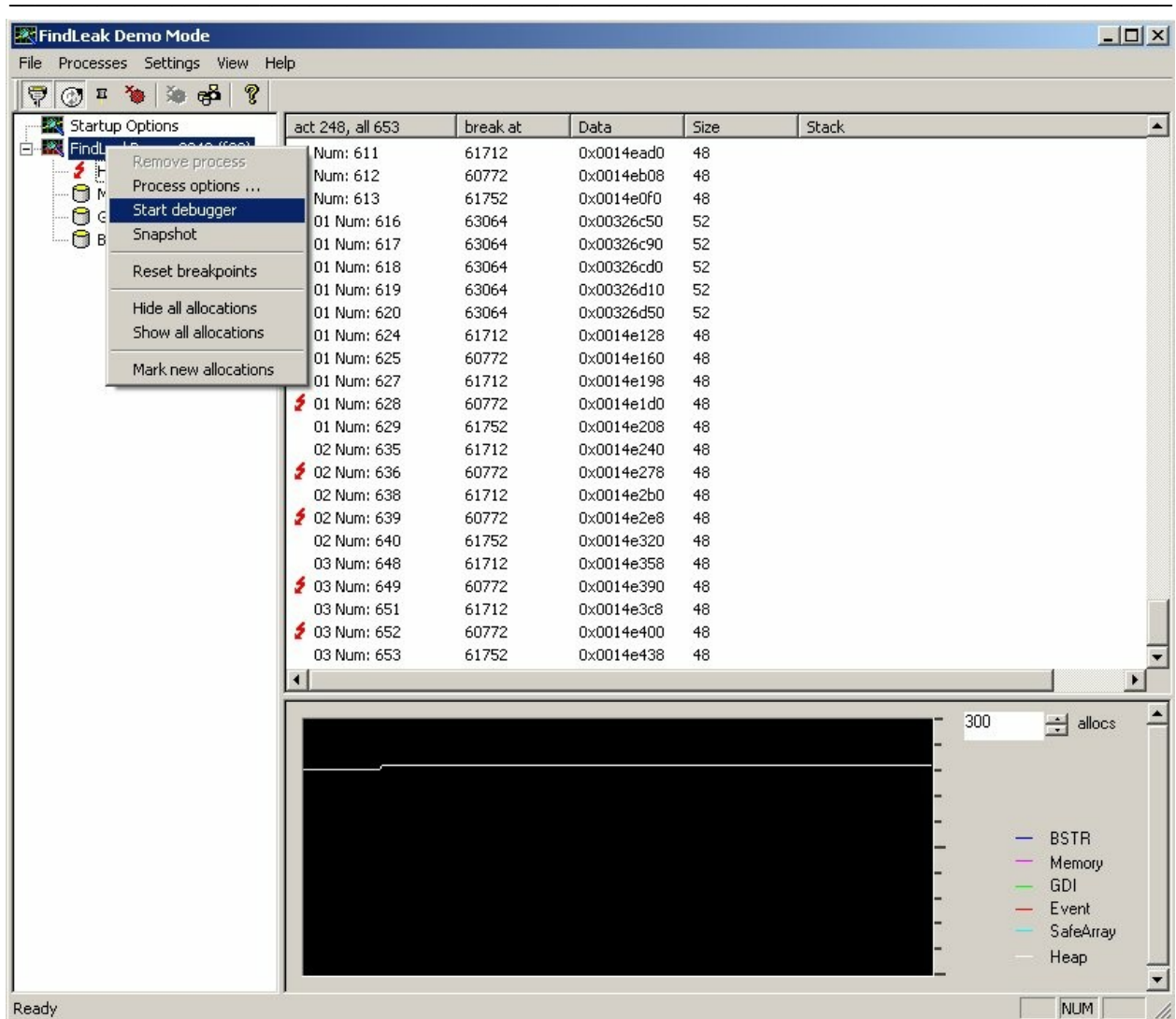
You can tag the corresponding source code for inspection by setting a break point. To do this, select one of the entries referring to the indicated leak and select "Break when allocated" in the context menu.



After having selected "Break when allocated", the tree view's "Heap" node shows a red flash, representing a break point. Also, each entry in the list view referring to selected memory location shows a red flash, too.



Right-click on "FindLeakDemo" in FindLeak's tree view and select "Start debugger" to invoke your development environment's debugger.



Clicking "AllocSysString" for another time will now halt FindLeakDemo at the marked position. With this it is possible for you to analyze FindLeakDemo's source code at the critical position.



## 7. A tour through FindLeak

After installation of FindLeak, a DLL called Applnit executes in the context of every starting process. If FindLeak is running, Applnit then attaches FindLeak to the starting process, otherwise the process executes as usual. By default, FindLeak attaches to every starting process. FindLeak monitors processes with the help of a so called Spy-DLL. The table below depicts the kind of resource allocations and de-allocations the Spy-DLL keeps track of.


**FindLeak is monitoring the following resources:**

BSTR	Functions for allocation and and release of BSTR in Oleaut32.dll
SafeArray	Functions for allocation and release of Safearrays in Oleaut32.dll
Memory	The malloc, calloc, realloc und free - functions in MSVCRT.dll and in MSVCRTD.dll as well as the GlobalAlloc and GlobalFree functions in Kernel32.dll
Event	CreateEvent and CloseHandle functions in Kernel32.dll
GDI	Functions for the creation and the release of GDI objects (except bitmaps) in the GDI32.dll
Heap	HeapAlloc and HeapFree

FindLeak's process view lists every process that is monitored by the Spy-DLL. Each allocation of the above listed resources causes a list entry in FindLeak's allocation view for that process. Upon de-allocation of a resource the corresponding list item gets removed from the allocation view. Thus, at any given point FindLeak lists every allocated resource of a process.

### FindLeak's main window

After being started, FindLeak's process view shows the start-up options. The allocation view on the right hand side only lists one item: „<Other>“. Double-click on „<Other>“ pops up the „Process control“ dialog. Here you can set options that apply for each newly started process. In addition you can set options that only apply to a certain process by entering a process specific entry through the context menu.


FindLeak lists all monitored processes in the process tree using their default icon. If there is no default icon, the  icon is used instead. Selecting a process in the process view, shows details about it's allocated resources in the allocation view. That is, a list of all used resources, the current number of allocations, and the maximum number of allocations. Furthermore, each process entry in the process view has sub-entries for the resource types heap, event, memory, and GDI. Selecting a resource type will show all allocated resources of that type in the allocation view. The table below depicts the allocation view's column layout when viewing a particular resource:

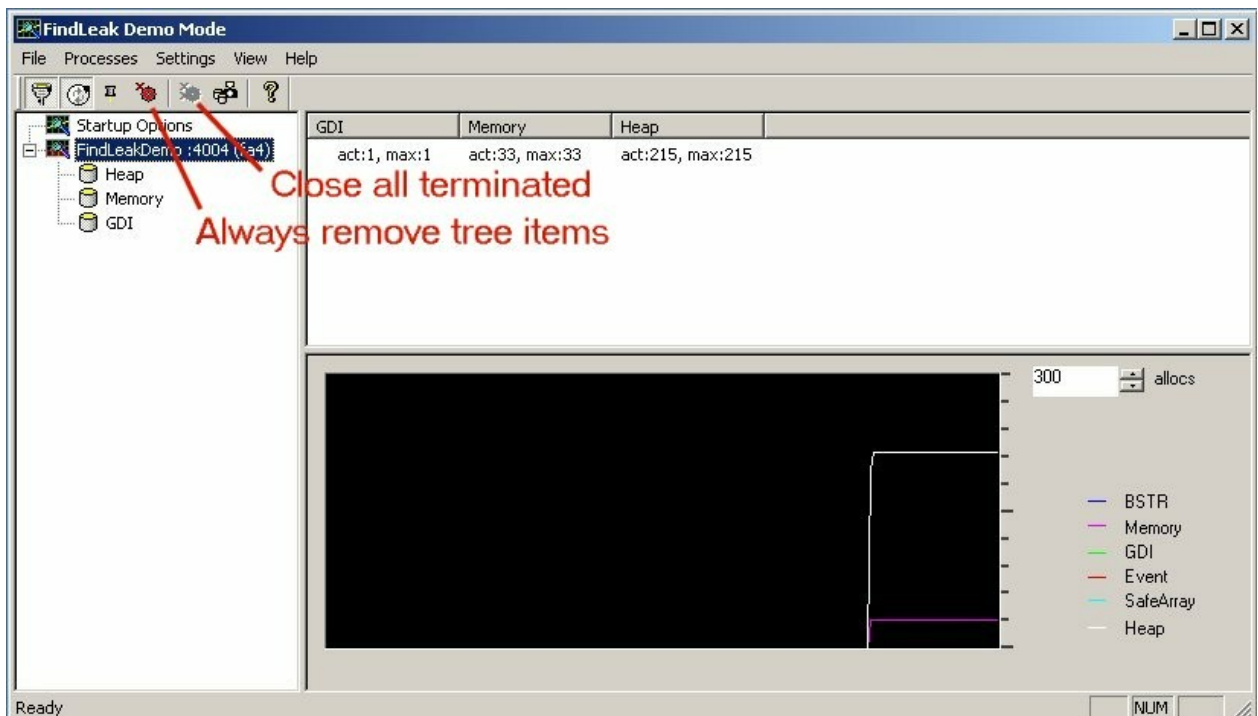
1.column	Every resource allocation can be identified by it's allocation number. This allocation number
----------	---



	consists of the cycle prefix, a separator, and the allocation count. The column header shows the currently selected type's number of allocations and the number of all allocations.
2. column	Shows a number generated by FindLeak. This number corresponds to the memory address of the code that allocated the resource.
3. column	Pointer or handle to the resource
4. column	Depending on the resource type: For „Memory“, „Heap“, „BSTR“ or „SafeArray“: the size of the allocated amount of memory. If „Event“: the event's name (if it is of the type „named event“). If „GDI“: the GDI type is.
5. column	If option „Display callstack“ is activated the callstack at time of allocation is displayed.

The graph view, below the allocation view, plots the number of allocations over time.

Once a process has terminated and freed all it's resources, it will be removed from FindLeak's process view. If not all resources were freed, the process remains in the process view and will be marked as terminated with a red  icon. A click on "Close all terminated" will remove all processes from the process view that are currently marked as terminated. FindLeak will automatically remove every terminated process, once „Always remove tree items“ has been selected.



## 8. Working with FindLeak

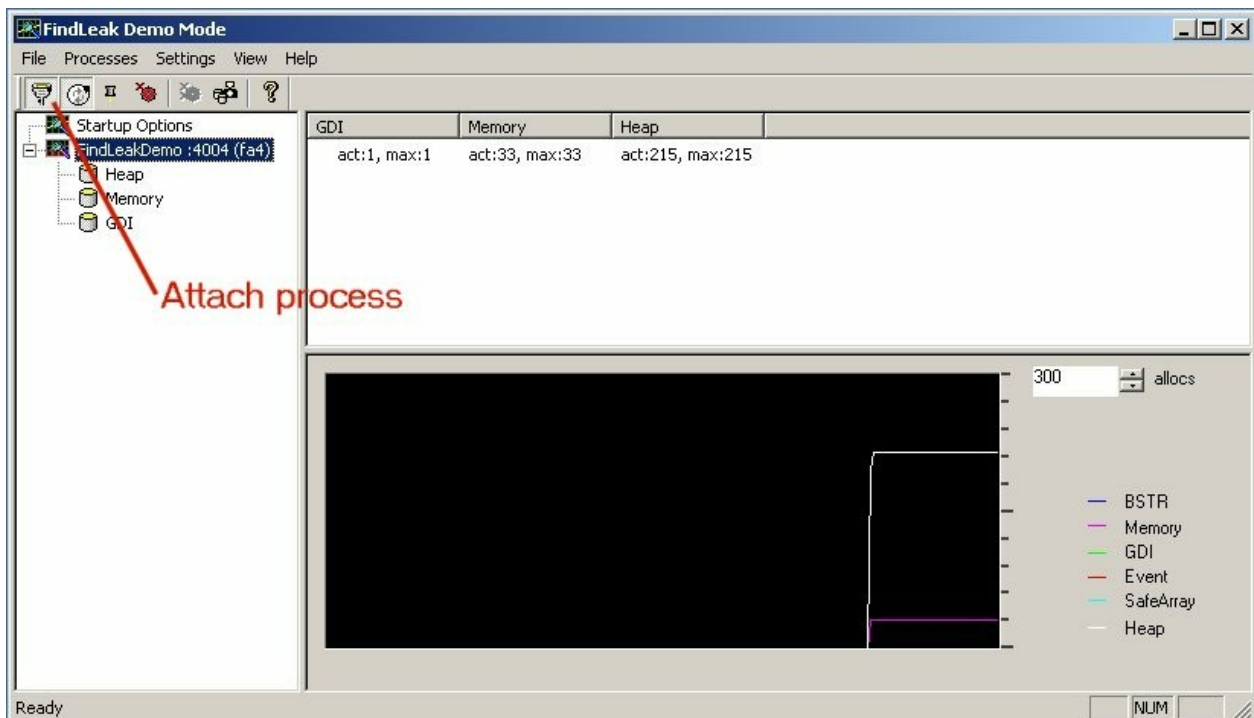
### Attaching the debugger

When working with FindLeak, using the debugger for inspecting code becomes a necessity at some point. There are several ways of attaching a debugger to the program, monitored by FindLeak:

- Start the debugger and attach it to the process (e.g. Build | Start Debug | Attach to Process in Microsoft Visual Studio 6.0 with SP5). Note, however, that you should disable FindLeak's "Attach process" feature before starting your debugger to prevent FindLeak from monitoring it, too.
- Directly attach the debugger by right-clicking the process in FindLeak's process view and selecting "Debug Process". This way, FindLeak will start the system's default debugger and attach it to the selected process (without trying to attach itself to debugger).

### Detecting of multiple resource de-allocations

FindLeak halts a monitored process, whenever it detects a de-allocation for a resource that has already been freed. If the process is running within a debugging environment, the debugger will break it's execution. If the process isn't being debugged, an appropriate message box will indicate the error. It is then possible to attach a debugger to the process



The memory resource in question might have been freed at two different places in your source code. Yet, FindLeak only shows where memory got de-allocated for the second time. For finding the first location, turn back to FindLeak and enable "show stack". With this, FindLeak will display the call stack of the first de-allocation within the debugger, once the process is stopped due to the same multiple resource de-allocation.

## Checking for memory leaks

The following three step procedure helps you to find memory leaks within a process.

### **First step:**

Attach FindLeak to the process and perform a test-run. The process should be exited in an orderly fashion. Any resources that are still left allocated will now be displayed in FindLeak.

Note, once a process has terminated, FindLeak might still report unfreed resources, such as resources that will be freed by the operating system, or resources held by the Spy-DLL.

It is not unusual for FindLeak to display unfreed resources after an application has terminated. The amount of resources left allocated should be the same for each run of the application. If the amount varies, your application most likely has a memory leak.

### **Second step:**

Monitor the number of allocations for different test scenarios in FindLeak allocation view. An increase in the number of allocations over time indicates a potential memory leak.

Make sure your test scenario always ends in the same application state. You might have to repeat the scenario several times, as caching mechanisms might interfere with the allocation overview.

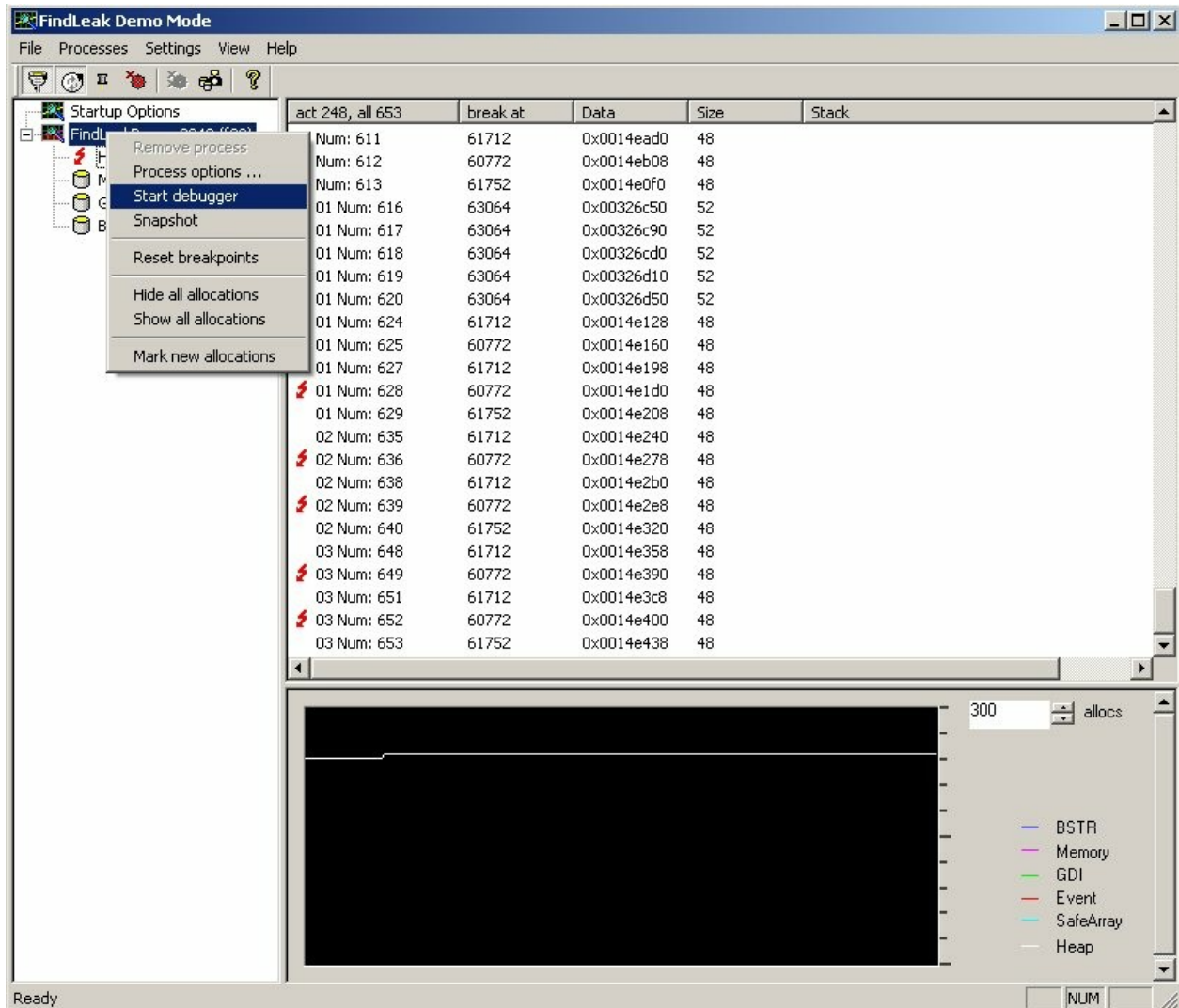
### **Third step:**

Once a test scenario indicates a memory leak, you can narrow down the leak with the help of the detailed resource view.

Select „Hide all allocations“ in the allocation view's context menu and repeat your test scenario. The allocation view will now only show your scenario's resource allocations. Right-click on an allocation entry and select „Mark new allocations“ in the popup-menu. Repeat your test scenario. The list of allocations will no be appended with new allocations, clearly marked by their allocation number. Also, all (non-marked) allocations of the previous repeat should now have disappeared. Any allocation that did not disappear indicates a leak. Please refer to the next chapter for information on how to find a leak's matching source code.

## Finding leaky code

Once a resource leak has been discovered with FindLeak, you can find the matching source code through your debugging environment. Please refer to section 8.1 on how to attach a debugger to your application. We recommend selecting "Start debugger" in the process view's popup menu.



After the debugger has been attached, set a breakpoint in FindLeak's allocation view by selecting "Break when allocated" from the popup menu of the leak entry. Repeat the test scenario. As soon as the marked resource is allocated the debugger breaks the applications execution. You can now examine in the debugger, where the resource was allocated within the application's source code.

## 9. Registering FindLeak

If you'd like to register your FindLeak demo, select "Register" in the FindLeak demo's "Help" menu. The screenshot below depicts the dialog that will appear:

**Register FindLeak**

You can order a full-version of FindLeak here. Compared to the demo version, the full version also allows to create and edit your own projects.  
The license is user- and hardware-dependent; this means that it is valid for the computer your demo version is currently installed on.  
To order a FindLeak license, click on the button "Order form". An order form is generated containing your user data; please fax a signed copy of this form to us.  
Detailed information on ordering FindLeak can be found on the form.

If you still have questions about ordering FindLeak, please send us an E-Mail to: [sales@findleak.de](mailto:sales@findleak.de)

Identification: 205 907 914 F04 235 5B5 CB4 774 484

Name:

Company: cjt Systemsoftware AG

If you received a license key, put it here.

### 1. Registering / Buying the full featured version

Click on „Order form“ to make FindLeak create and show the order form. Print the order form and fill in the blanks. You can send the order form by mail or by fax. After receiving your order form we will send you a license key. This key entitles you to use FindLeak exclusively on the computer, for which this identification has been created.

### 2. Entering the license key

As soon as you have received the license key you can turn your copy of the FindLeak demo into a full featured version. Start the FindLeak demo and select the "Help" menu's "Register" item. Enter the license key into the license key field. Click "OK" and restart FindLeak. FindLeak has now turned into an unrestricted version.

## 10.Problems und Solutions

### ***No call stack visible***

„Step over“ a few instructions. In most cases the debugger will show the call stack after a few steps.

### ***FindLeak is only showing code within system DLLs***

This usually happens with calls to the Oleaut32 DLL, when it allocates memory to create interfaces. The problem resides mostly within the automation function, that can't be debugged. It might be helpful to have a closer look on the allocated resource to get information on which function should be called. The memory allocated by the Oleaut32 DLL is mostly freed correctly.

The way FindLeak finds the code which is allocating a resource might not always be clear. FindLeak might expect similar part's of source code to be equal. Just keep on debugging or running the application. There should be another break at a different source code where the resource is really allocated.

## 11.Frequently asked questions

### ***How does the free demo version of FindLeak differ from the full version?***

FindLeak's demo version is restricted in such that it allows you to only work with the included demo project. None of FindLeak's other features are restricted.

### ***Why does FindLeak indicate that the activation failed, although I entered a valid license key?***

FindLeak's licensing mechanism generates a unique identification key based on the hardware configuration of your computer. We then use this identification key to generate a license key for activating your version of FindLeak. FindLeak will only work on the computer, for which the identification key was generated. Note, however, that a fundamental change in you computer's hardware setup might change your identification key and therefore invalidate your license key. If that happens, please contact our sales department via telephone ( +49 (7244) 73 41 13 ) or per mail: [sales@findleak.de](mailto:sales@findleak.de) in order to request a new license key.

### ***Is it possible to concurrently use FindLeak and another leak finding tool?***

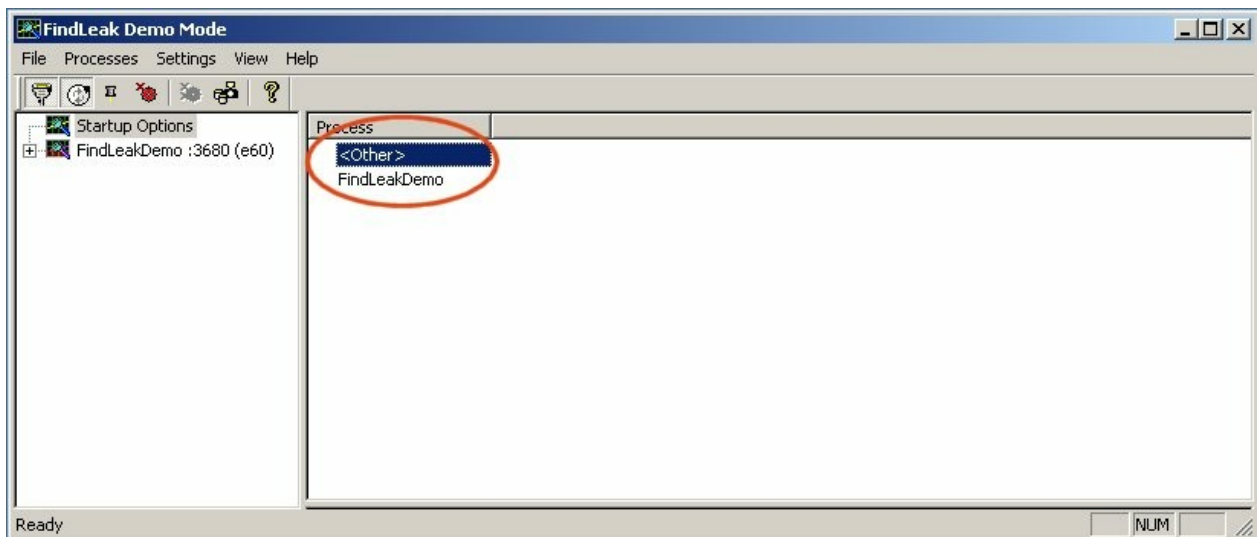
No. Memory debugging tools typically introduce their own memory management routines that rely on getting directly called out of the monitored application. A second tool for finding leaks would interfere with FindLeak and lead to unpredictable behavior.

***While being monitored with FindLeak, my application seems to have increased memory requirements. What is the reason for that?***

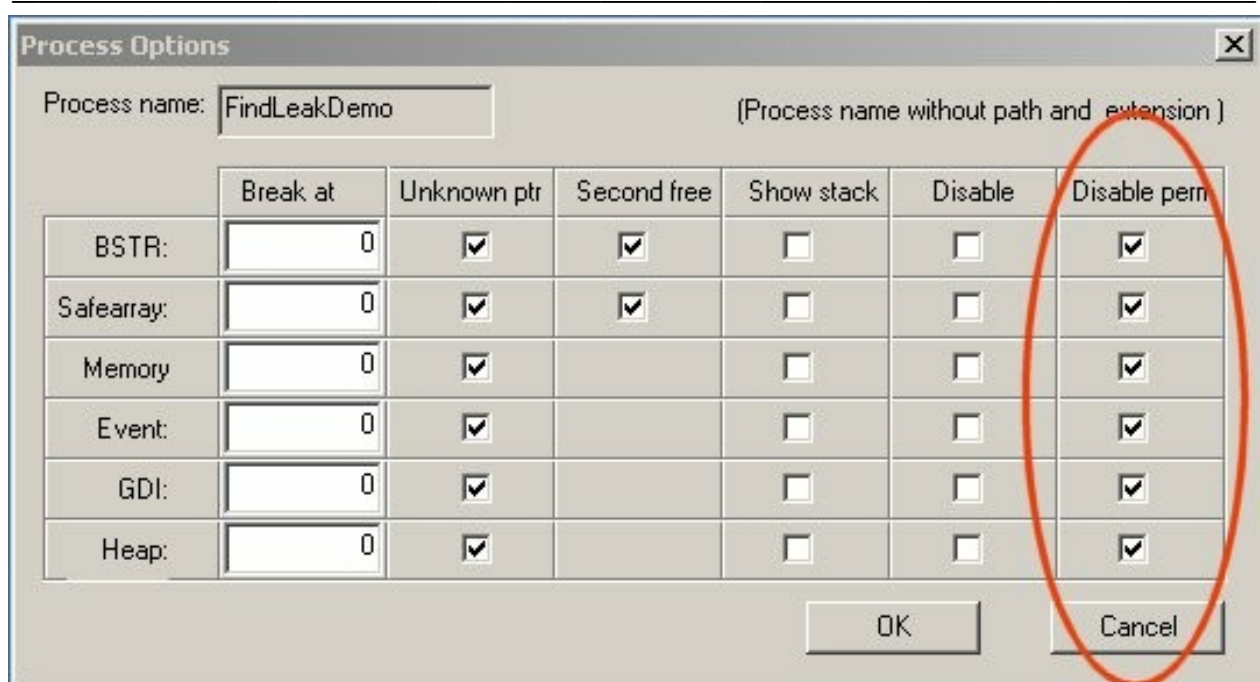
While monitoring a process, FindLeak requires additional memory for keeping track of resource (de-) allocations and therefore increases the memory usage of the process examined. Make sure that FindLeak is not active if you want to measure your process' memory requirements.

***FindLeak attaches itself onto each starting process. However I would like it to only examine a certain process.***

You can set FindLeak up for only attaching to processes matching a certain name. Please click on "Startup Options" in the tree view on the left hand side of FindLeak's main window. Then double-click on the "<OTHER>" entry in the list view on the right hand side (see below).



In the now appearing pop up window you can deactivate process monitoring for all kinds of memory types by checking the appropriate check box "Disable perm" (disable permanently) (see next fig.)



Process Options

Process name:  (Process name without path and extension)

	Break at	Unknown ptr	Second free	Show stack	Disable	Disable perm
BSTR:	<input type="text" value="0"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Safearray:	<input type="text" value="0"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Memory	<input type="text" value="0"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Event:	<input type="text" value="0"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
GDI:	<input type="text" value="0"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Heap:	<input type="text" value="0"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

OK Cancel

Choosing this option causes that immediately no more processes are included automatically into monitoring by FindLeak. If you would monitor these processes again, insert for each process a new entry into the process list. Start thereafter FindLeak again.

***When working with FindLeak, my system's performance degrades. Is there any way to improve the performance?***

Your application will allocate significantly more memory while being monitored with FindLeak. This can cause your system to start swapping and thus degrade the overall performance.

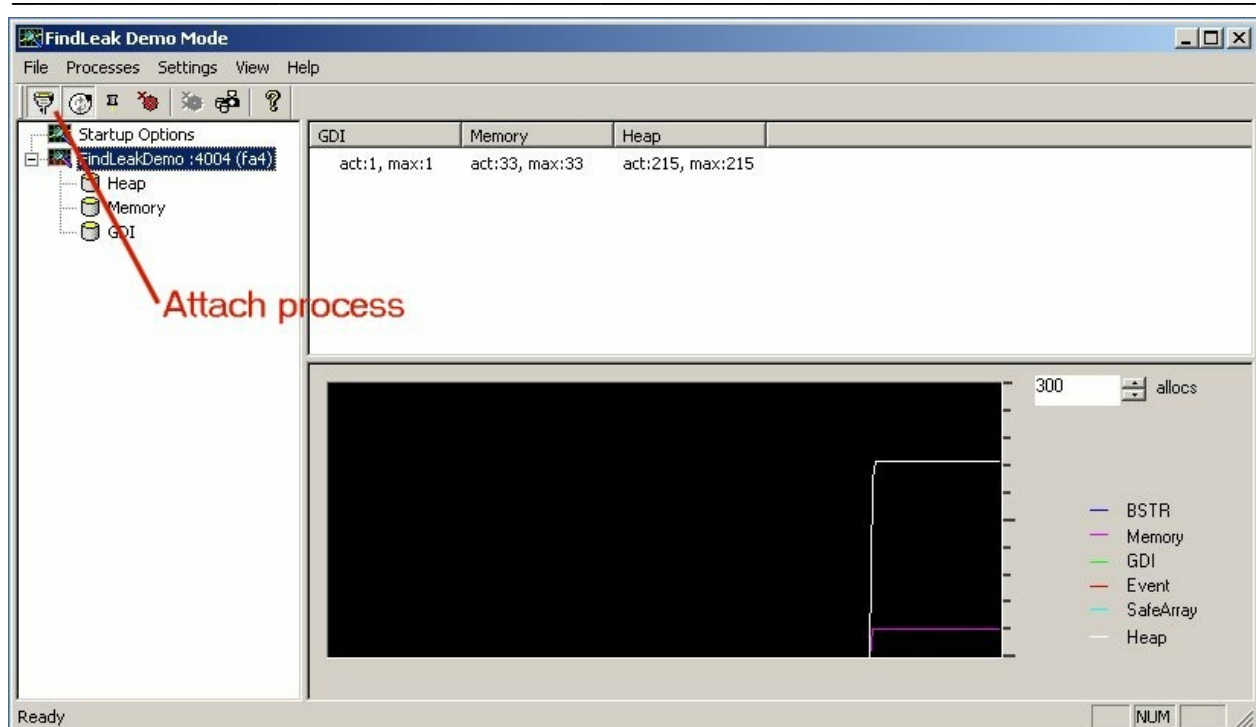
The memory requirements can be lowered by finding leaks with the following two step approach:

- (1) Disable monitoring for "heap" errors and remove all leaks for the remaining error types.
- (2) Only monitor and remove "heap" related leaks .

***How can I exclude starting processes from being monitoring by FindLeak?***

FindLeak stops attaching to newly started processes once the "Attach process" mode has been disabled. This can be done by either clicking the "Attach process" button (see screenshot below) or by making sure "Attach Process" entry in the "Settings" menu is not checked.



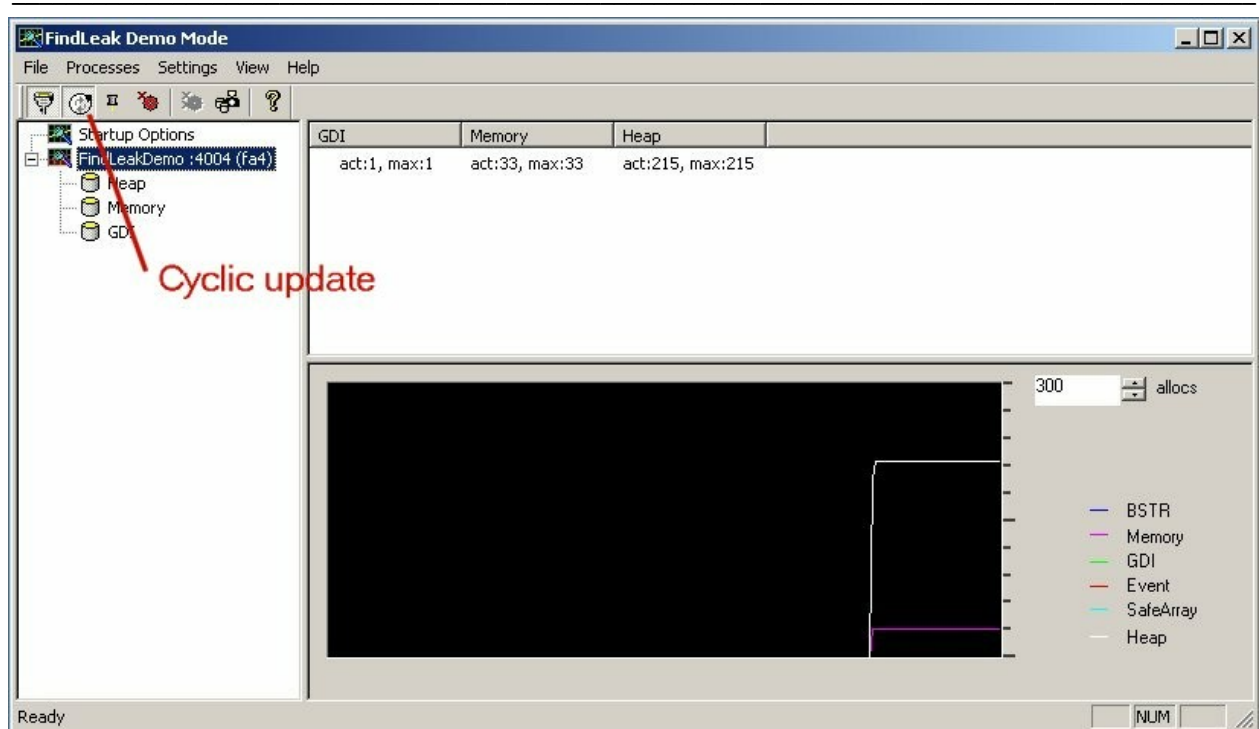


You can revert to the "Attach Process" mode by checking the Settings | Attach Process menu entry.

### ***How can I accelerate my work with FindLeak?***

You can speed up your work with FindLeak by either disabling the "Cyclic update" option (a) or by only focusing on certain types of memory errors (b).

(a) By default, FindLeak periodically refreshes the list of memory allocations on the right hand side of the screen. However, redrawing this list can be time consuming if the number of allocation entries is high. Click on the "Cyclic update" button (as depicted in the screenshot below), or make sure the Settings | Cyclic update menu entry is not checked to disable periodic updates of the allocations list. You can always reactivate periodic updates by again clicking the above mentioned button/menu.



(b) Improving FindLeak's performance by focusing on certain types of memory errors works as follows:

Divide the investigation of memory allocations into two steps:

- Disable monitoring for "heap" errors and remove all leaks for the remaining error types.
- Only monitor and remove "heap" related leaks.

## 12.Menu Reference

### „Process“ Menu

#### **Snap shot**

Stores current processes number of allocated resources.

#### **Close all terminated**

Removes all terminated processes from FindLeak's process view.

#### **Reset breakpoints**

Resets all breakpoints of the current process.

#### **Hide all**

Hides all allocations of the current process.

#### **Show all**

Sets all allocations of the current process to visible.

#### **Mark new allocations**

Marks a new allocation cycle by incrementing the allocation prefix.

#### **Attach to process...**

Attaches FindLeak to a running process.

### „Settings“ Menu

#### **Auto-Attach New Processes**

If "Auto-Attach New processes" is checked, FindLeak automatically attaches itself to any newly started process.

#### **Cyclic update**

Periodically refreshes FindLeak's allocation view, if checked. If not checked, you can manually refresh the allocation view by clicking onto a resource in the process view.

Cyclic updates can slow down the monitored process if the number of allocations is high.

#### **Always on top**

If checked, FindLeak will always stay on top of all windows.

#### **Auto-Remove Terminated Processes**

If checked, FindLeak removes every terminated process from FindLeak's process view, even if the process still holds allocated resources. If not checked, a terminated process will only be removed, if it has freed all of its resources.

#### **Cyclic mark...**

Marks new allocations in a user defined cycle.

#### **Symbol path...**

Shows a dialog for setting the symbol path of symbols downloaded from the Microsoft symbol server.

#### **Select language...**

This menu is for changing the user interface language.

## Context menu of FindLeak's process view

This context menu is shown if you rightclick on a process that is shown in FindLeak's process view. These processes can have different icons representing their:



: the process is active and is monitored by FindLeak.



: the process has been monitored by FindLeak, but is not active anymore

### **Attach**

Re-attaches FindLeak to the selected process process.

### **Remove process**

Removes an inactive (terminated) process from FindLeak's process view.

### **Options**

Brings up a dialog that is used to select monitoring options for that process.

### **Start debugger**

Starts the system's default debugger and attaches it to the current process. Will not attach FindLeak to the debugger.

### **Snap shot**

Stores the count of all allocated resources of the highlighted process is stored.

### **Modules**

This menu shows a list of all modules as loaded by the selected process.

### **Reset breakpoints**

Resets all breakpoints (red flash symbol) of the current process.

### **Hide all**

Hides all allocations of the current process.

### **Show all**

Sets all allocations of the current process to visible.

### **Mark new**

All resources that get allocated after this button is clicked are numbered. Everytime this button is clicked the numbering is increased.



### **Resource type item:**

### **Reset breakpoints**

Resets all breakpoints (red flash symbol) of the current allocation type.

### **Hide all allocations**

Hides all allocations of the current resource type.

### **Show all allocations**

Sets all allocations of the current resource type to visible.

### **Mark new allocations**

All allocations of the current resource type get allocated after this button is clicked are numbered. Every time this button is clicked the numbering is increased.

### **Contextmenu of FindLeak's allocation view**

### ***Startup options***

#### ***Add options ...***

Add startup options for a specific process

#### ***Remove options ...***

Remove a process' startup options from FindLeak's options list.

#### ***Change options ...***

Change a process' startup options.

### ***Snap shot***

Stores the total number of allocated resources for the current process.

### ***Details ...***

Shows a memory window for the selected resource. For the resource types heap and memory only.

### ***Show stack ...***

Shows the call stack for the current process, if checked.

### ***Break when allocated***

Adds a breakpoint to the selected item's address. A red flash appears.

### **Reset breakpoints**

Resets all breakpoints (red flash symbol) of the current process.

### **Hide all allocations**

Hides all allocations of the current resource type.

### **Show all allocations**

Shows all allocations of the current resource type.

### **Mark new allocations**

All allocations of the current resource type get allocated after this button is clicked are numbered.

Everytime this button is clicked the numbering is increased.

## **Dialogs**

### **Call Stack**

FindLeak can show the call stack of the current process, as depicted in the screenshot below.



Adresse	Module	Funktion	Datei	Zeile
774CDE7B	ole32.dll	StringFromGUID2		
774D2A46	ole32.dll	CoInitialize		
004012B6	FindLeakDe...	CFindLeakDemoApp::InitATL	FindLeakDemo....	145
00401102	FindLeakDe...	CFindLeakDemoApp::InitInstance	FindLeakDemo....	46
73D3CF74	MFC42.DLL	Ordinal1576		
7C816D4F	kernel32.dll	RegisterWaitForInputIdle		

You can match an address of the call stack to a particular function with the help of your debugger, provided that the debugger is already attached to the application

Copy the stack address from FindLeak's stack window into the clipboard. Switch to the debugger and make sure the disassembly window is open. Open the Jump to the address dialog (Strg – G in Microsoft Visual Studio 6), paste the address out of the clipboard, and press the "Go To" button. In Microsoft Visual Studio you can now display the corresponding source code with "Go to source".

### **What stack addresses are of interest in case of debugging?**

Most memory errors are located within the process' code base. Accordingly, these errors happen in the current process's (or DLL's) address range. You can find out about it's address range with the help of your debugger (in Microsoft Visual Studio: click "Debug" and "Modules").

## Process options

	Break at	Unknown ptr	Second free	Show stack	Disable	Disable perm
BSTR:	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Safearray:	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Memory	0	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Event:	0	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
GDI:	0	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Heap:	0	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

With the process options you can configure how FindLeak should monitor a specific process. The specific set of options is read when FindLeak attaches itself to a process. You can always change a process' set of options by selecting "process options ..." in the process' context menu. If there are no specific options for a process FindLeak uses the default options.

### **unknown Ptr**

Enable this option to check for pointer inconsistencies within FindLeak. Upon detection, FindLeak presents a message box „FindLeak failed ...“. This notifies you that you can't rely anymore on FindLeak's output.

### **second free**

Sets FindLeak to check for multiple de-allocations. See chapter 5 for more details.

### **show stack**

Saves a call stack for every allocation. This option is unnecessary for most test scenarios, as it is possible to find source code belonging to a resource leaks in other ways.

Please refer to chapter 8.

Warning: The 'show stack' option considerably decreases FindLeak's (and your application's) performance.

### **disable**

Disables monitoring for that specific resource type.

### **break at**

Enter the address of the resource that should trigger a debug break. Please refer to "Break when allocated".