Memory Leak Detector

Last Updated: February 5, 2013

The Memory Leak Detector feature is a tool that can be used to detect memory leaks on a router that is running Cisco IOS software. The Memory Leak Detector feature is capable of finding leaks in all memory pools, packet buffers, and chunks.

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information for Memory Leak Detector.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Memory Leak Detector

- You should have at least a basic familiarity with the Cisco IOS environment and the command-line interface.
- You should have at least a minimal configuration running on your system.

Restrictions for Memory Leak Detector

- You must have your network up and running, with Cisco IOS Release 12.2 or a later release installed.
- Some of the Cisco IOS configuration commands are only available on certain router platforms, and the command syntax may vary on different platforms.

Information About Memory Leak Detector

Memory Leaks

Memory leaks are static or dynamic allocations of memory that do not serve any useful purpose. Although technology is available for detection of leaks among statically allocated memory, in this document the focus is on memory allocations that are made dynamically.

Memory Leak Detection

From the detection point of view, leaks among the dynamically allocated memory blocks can be classified into the following three types:

- Type 1 leaks have no references. These blocks of memory can not be accessed.
- Type 2 leaks are part of one or more cycles of allocations but none of the blocks in these cycles is accessible from outside of the cycles. Blocks within each cycle have references to other elements in the cycle(s). An example of a Type 2 leak is a circular list that is not needed anymore. Though individual elements are reachable, the circular list is not reachable.
- Type 3 leaks are accessible or reachable but are not needed, for example, elements in data structures that are not needed anymore. A subclass of Type 3 leaks are those where allocations are made but never written to. You can look for these subclass leaks using the **showmemorydebugreferenceunused** command.

The Memory Leak Detector feature provides the technology to detect Type 1 and Type 2 memory leaks.

The Memory Leak Detector feature works in the following two modes:

- Normal mode--Where memory leak detector uses memory to speed up its operations.
- Low memory mode--Where memory leak detector runs without attempting to allocate memory.

Low memory mode is considerably slower than the normal mode and can handle only blocks. There is no support for chunks in low memory mode. Low memory mode is useful when there is little or no memory available on the router.

The memory leak detector has a simple interface and can be invoked by the command line interface (CLI) at any time to get a report of memory leaks. For testing purposes, you can perform all tests, then invoke memory leak detector to get a report on leaks. If you are interested only in leaks generated by your test cases alone, memory leak detector has an incremental option, which can be enabled at the start of testing. After testing completes, you can get a report on only the leaks that occurred after the incremental option was enabled.

To reduce false alarms, it is mandatory that memory leak detector be invoked multiple times and that only leaks that consistently appear in all reports be interpreted as leaks. This is especially true for packet buffer leaks.

Note When submitting defects based on the reports of memory leak detector, please add “memleak-detection” to the attribute field of the defect report.

Danger Executing memory leak detection commands on a device with a serious memory leak issue may cause loss of connectivity.

How to Use Memory Leak Detector

Displaying Memory Leak Information

To display detected memory leak information, complete the task in this section:

SUMMARY STEPS

1. **enable**
2. **show memory debug leaks** [**chunks** | **largest** | **lowmem** | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**show memory debug leaks** [**chunks** | **largest** | **lowmem** | **summary**] **Example:** `Router# show memory debug leaks chunks`	Invokes normal mode memory leak detection and displays detected memory leaks. Optional keywords are as follows: • **chunks** --Invokes normal mode memory leak detection and displays detected memory leaks in chunks. • **largest** --Invokes memory leak detection and displays the top ten leaking allocator_pcs and total amount of memory that they have leaked. Additionally, each time this command is invoked it remembers the previous invocation's report and compares it to the current invocation's report. • **lowmem** --Invokes low memory mode memory leak detection and displays detected memory leaks. The amount of time taken for analysis is considerably greater than that of normal mode. The output for this command is similar to the **showmemorydebugleaks** command. • **summary** --Invokes normal mode memory leak detection and displays detected memory leaks based on allocator_pc and then on the size of the block.

Setting the Memory Debug Incremental Starting Time

To set the starting time for incremental analysis of memory leaks, complete the task in this section:

SUMMARY STEPS

1. **enable**
2. **set memory debug incremental starting-time**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**set memory debug incremental starting-time** **Example:** `Router# set memory debug incremental starting-time`	Sets the starting time for incremental analysis to the time when the command is issued. When the starting time is set, only memory allocated after the starting time will be considered for reporting as leaks.

Displaying Memory Leak Information Incrementally

To display memory leak information after a starting time has been established, complete the tasks in this section:

SUMMARY STEPS

1. **enable**
2. **set memory debug incremental starting-time**
3. **show memory debug incremental** {**allocations** | **leaks** [**lowmem**] | **status**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**set memory debug incremental starting-time** **Example:** `Router# set memory debug incremental starting-time`	Sets the starting time for incremental analysis to the time when the command is issued.
Step 3	**show memory debug incremental** {**allocations** | **leaks** [**lowmem**] | **status**} **Example:** `Router# show memory debug incremental allocations` **Example:**	• **allocations** --Displays all the memory blocks that were allocated after the issue of a **setmemorydebugincrementalstarting-time** command. The displayed memory blocks are just memory allocations, they are not necessarily leaks. • **leaks** --Displays output similar to the **showmemorydebugleaks** command, except that it displays only memory that was leaked after the issue of a **setmemorydebugincrementalstarting-time** command. • **lowmem** --Forces memory leak detection to work in low memory mode. The output for this command is similar to the **showmemorydebugleaks** command, except that it displays only memory that was leaked after the issue of a **setmemorydebugincrementalstarting-time** command. ◦ In low memory mode, the analysis time is considerably greater than it is in normal mode. ◦ You can use this command when you already know that normal mode memory leak detection will fail (perhaps by an unsuccessful previous attempt to invoke normal mode memory leak detection). • **status** --Displays whether a starting point for incremental analysis has been set and the elapsed time since then.

Examples for Memory Leak Detector

- Example show memory debug leaks, page 6
- Example show memory debug leaks chunks, page 6

Example show memory debug leaks

The following example shows output from the **showmemorydebugleaks** command with no optional keywords specified:

```
Router# show memory debug leaks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28       80 60616750  -2   Init
62DABD78       80 606167A0  -2   Init
62DCF240       88 605B7E70  -2   Init
62DCF298       96 605B7E98  -2   Init
62DCF2F8       88 605B7EB4  -2   Init
62DCF350       96 605B7EDC  -2   Init
63336C28      104 60C67D74  -2   Init
63370D58       96 60C656AC  -2   Init
633710A0      304 60C656AC  -2   Init
63B2BF68       96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
```

The table below describes the significant fields shown in the display.

Table 1 show memory debug leaks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.

Example show memory debug leaks chunks

The following example shows output from the **showmemorydebugleakschunks** command:

```
Router# show memory debug leaks chunks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
Address  Size  Parent   Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
```

```
Address  Size  Parent   Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28        80 60616750  -2   Init
62DABD78        80 606167A0  -2   Init
62DCF240        88 605B7E70  -2   Init
62DCF298        96 605B7E98  -2   Init
62DCF2F8        88 605B7EB4  -2   Init
62DCF350        96 605B7EDC  -2   Init
63336C28       104 60C67D74  -2   Init
63370D58        96 60C656AC  -2   Init
633710A0       304 60C656AC  -2   Init
63B2BF68        96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
Chunk Elements:
Address  Size  Parent   Name
62D80DA8    16 62D7BFD0 (Managed Chunk )
62D80DB8    16 62D7BFD0 (Managed Chunk )
62D80DC8    16 62D7BFD0 (Managed Chunk )
62D80DD8    16 62D7BFD0 (Managed Chunk )
62D80DE8    16 62D7BFD0 (Managed Chunk )
62E8FD60   216 62E8F888 (IPC Message He)
```

The table below describes the significant fields shown in the display.

Table 2 show memory debug leaks chunks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.
Size	(Chunk Elements) Size of the leaked element (bytes).
Parent	(Chunk Elements) Parent chunk of the leaked chunk.
Name	(Chunk Elements) The name of the leaked chunk.

Example show memory debug leaks largest

The following example shows output from the **showmemorydebugleakslargest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
                PCI memory
Alloc_pc    total leak size
                I/O memory
Alloc_pc    total leak size
                Processor memory
Alloc_pc    total leak size
608D2848    32776     inconclusive
608D2FD8    32776     inconclusive
```

```
60C656AC    288       inconclusive
60C67D74    48        inconclusive
605B7E98    40        inconclusive
605B7EDC    40        inconclusive
60C659D4    40        inconclusive
605B7E70    32        inconclusive
605B7EB4    32        inconclusive
60616750    24        inconclusive
```

The following example shows output from the second invocation of the **showmemorydebugleakslargest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
               PCI memory
Alloc_pc    total leak size
               I/O memory
Alloc_pc    total leak size
               Processor memory
Alloc_pc    total leak size
608D2848    32776
608D2FD8    32776
60C656AC    288
60C67D74    48
605B7E98    40
605B7EDC    40
60C659D4    40
605B7E70    32
605B7EB4    32
60616750    24
```

Example show memory debug leaks summary

The following example shows output from the **showmemorydebugleakssummary** command:

```
Router# show memory debug leaks summary
Adding blocks for GD...
               PCI memory
Alloc PC        Size     Blocks      Bytes    What
               I/O memory
Alloc PC        Size     Blocks      Bytes    What
               Processor memory
Alloc PC        Size     Blocks      Bytes    What
0x605B7E70 0000000032 0000000001 0000000032    Init
0x605B7E98 0000000040 0000000001 0000000040    Init
0x605B7EB4 0000000032 0000000001 0000000032    Init
0x605B7EDC 0000000040 0000000001 0000000040    Init
0x60616750 0000000024 0000000001 0000000024    Init
0x606167A0 0000000024 0000000001 0000000024    Init
0x608D2848 0000032776 0000000001 0000032776    Audit Process
0x608D2FD8 0000032776 0000000001 0000032776    Audit Process
0x60C656AC 0000000040 0000000001 0000000040    Init
0x60C656AC 0000000248 0000000001 0000000248    Init
0x60C659D4 0000000040 0000000001 0000000040    Init
0x60C67D74 0000000048 0000000001 0000000048    Init
```

The table below describes the significant fields shown in the display.

Table 3 show memory debug leaks summary Field Descriptions

Field	Description
Alloc PC	Address of the system call that allocated the block.
Size	Size of the leaked block.

Field	Description
Blocks	Number of blocks leaked.
Bytes	Total amount of memory leaked.
What	Name of the process that owns the block.

Example show memory debug incremental allocations

The following example shows output from the **showmemorydebugincremental** command when entered with the **allocations** keyword:

```
Router# show memory debug incremental allocations
Address    Size   Alloc_pc  PID  Name
62DA4E98      176 608CDC7C  44   CDP Protocol
62DA4F48       88 608CCCC8  44   CDP Protocol
62DA4FA0       88 606224A0  3    Exec
62DA4FF8       96 606224A0  3    Exec
635BF040       96 606224A0  3    Exec
63905E50      200 606A4DA4  69   Process Events
```

Example show memory debug incremental status

The following example shows output from the **showmemorydebugincremental** command entered with the **status** keyword:

```
Router# show memory debug incremental status
Incremental debugging is enabled
Time elapsed since start of incremental debugging: 00:00:10
```

Additional References

The following sections provide references related to Memory Leak Detector.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS configuration commands	*Cisco IOS Configuration Fundamentals Command Reference*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/public/support/tac/home.shtml

Feature Information for Memory Leak Detector

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4 Feature Information for Memory Leak Detector

Feature Name	Releases	Feature Information
Memory Leak Detector	12.3(8)T1 12.2(25)S	The Memory Leak Detector feature is a tool that can be used to detect memory leaks on a router that is running Cisco IOS software. The Memory Leak Detector feature is capable of finding leaks in all memory pools, packet buffers, and chunks.