Intel® Inspector XE 2013

Memory Checker
Thread Checker
Static Analysis
Pointer Checker

Code the Future

Additional Material

Intel® Inspector XE – Memory and Thread Checker / Debugger

Intel Inspector XE:

- Product page – overview, features, FAQs…
- Training materials – movies, tech briefs, documentation…
- Evaluation guides – step by step walk through
- Case studies
- Support – forums, secure support…
- Set up static analysis: C, C++ and Fortran

More Analysis Tools:

- Intel® Advisor XE – threading prototyping tool for architects
- Intel® VTune™ Amplifier XE – performance profiler

Intel Software Development Products

Optimization Notice

Correctness tools increase ROI by 12%-21%

Cost Factors – Square Project Analysis
CERT: U.S. Computer Emergency Readiness Team, and Carnegie Mellon CyLab
NIST: National Institute of Standards & Technology : Square Project Results

Size and complexity of applications is growing

Correctness tools find defects during development prior to shipment

Reworking defects is 40%-50% of total project effort

Reduce time, effort, and cost to repair

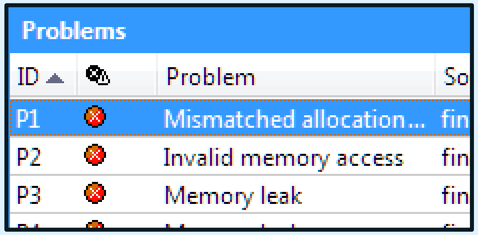
Find errors earlier when they are less expensive to fix

Deliver More Reliable Applications

Intel® Inspector XE and Intel® Parallel Studio XE family of suites

Dynamic Analysis

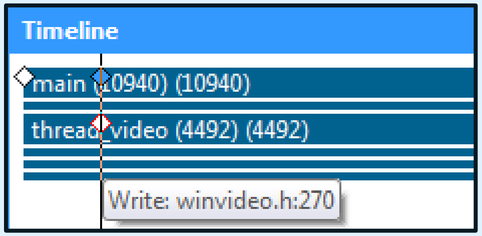
Memory Errors



Problems

ID		Problem	So
P1	●	Mismatched allocation…	fin
P2	●	Invalid memory access	fin
P3	●	Memory leak	fin

Threading Errors



Timeline

main (10940) (10940)

thread video (4492) (4492)

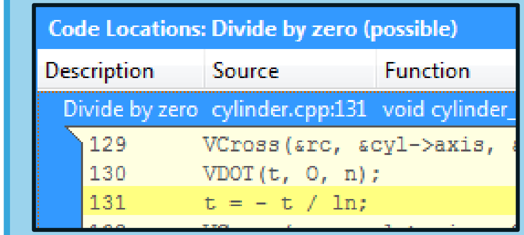
Write: winvideo.h:270

Intel® Inspector XE alone

Intel Inspector XE dynamically instruments & runs the application and watches for errors. Use any build, any compiler (debug build is best).

Static Analysis

Code & Security Errors



Code Locations: Divide by zero (possible)

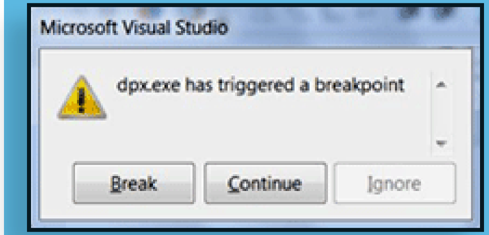
Description	Source	Function
Divide by zero	cylinder.cpp:131	void cylinder_

```
129        VCross(&rc, &cyl->axis, &
130        VDOT(t, O, n);
131        t = - t / ln;
```

Intel compiler inspects source.
Use any compiler for production.

Pointer Checker NEW

Pointer Errors



dpx.exe has triggered a breakpoint

Intel compiler run time checks.
Use any compiler for production.

Added bonus features in **Intel® Parallel Studio XE** suites

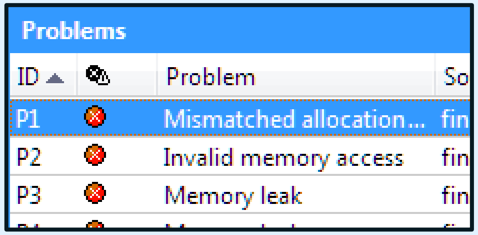
Static Analysis & Pointer Checker are only available in the Parallel Studio XE family of suites. Not sold separately.

Deliver More Reliable Applications

Intel® Inspector XE and Intel® Parallel Studio XE family of suites

Dynamic Analysis

Memory Errors

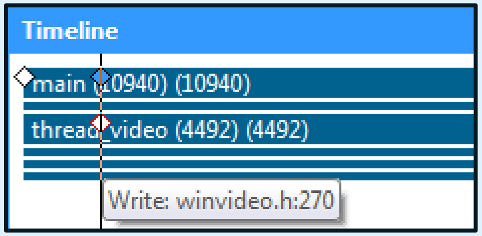


Problems

ID ▲		Problem	So
P1	●	Mismatched allocation…	fin
P2	●	Invalid memory access	fin
P3	●	Memory leak	fin

- Invalid Accesses
- Memory Leaks
- Uninit. Memory Accesses

Threading Errors

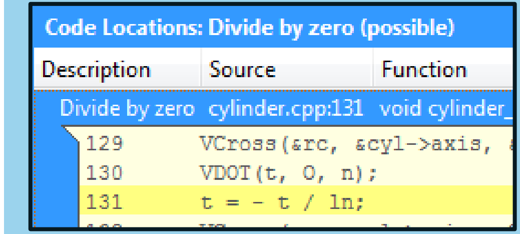


- Races
- Deadlocks
- Cross Stack References

- Multiple tools
- One common user interface
- Easy workflow for developers
- Windows & Linux

Static Analysis

Code & Security Errors



Code Locations: Divide by zero (possible)

Description	Source	Function
Divide by zero	cylinder.cpp:131	void cylinder_

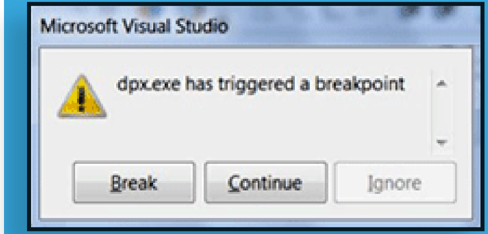
```
129      VCross(&rc, &cyl->axis, &
130      VDOT(t, O, n);
131      t = - t / ln;
```

- Buffer over/under flows
- Incorrect pointer usage
- Over 250 error types…

Pointer Checker

NEW

Pointer Errors



Microsoft Visual Studio

dpx.exe has triggered a breakpoint

- Out of bounds accesses
- Dangling pointers

Find errors earlier with less effort

Static Analysis & Pointer Checker are only available in the Parallel Studio XE family of suites. Not sold separately.

Dynamic Analysis Finds Memory & Threading Errors

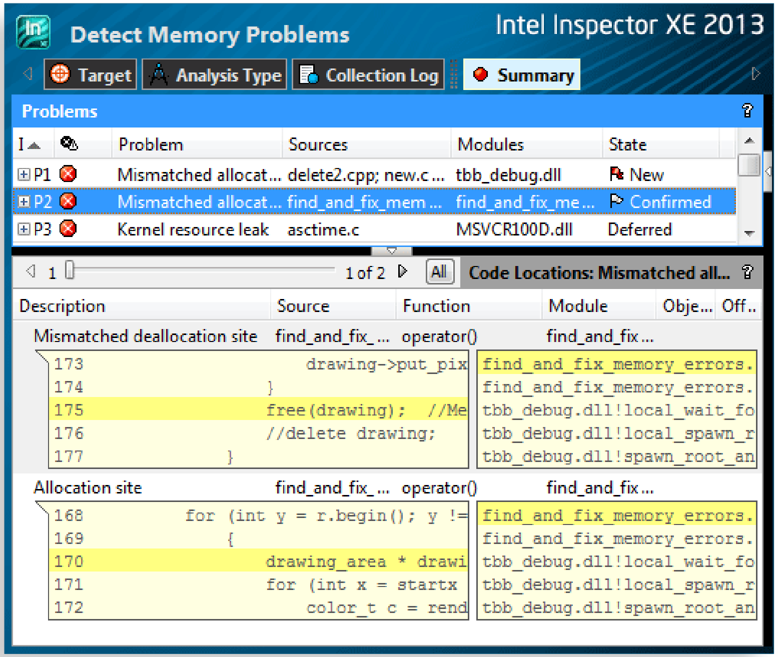
Intel® Inspector XE 2013

Find and eliminate errors

- Memory leaks, invalid access…
- Races & deadlocks
- C, C++, C#, F# and Fortran (or any mix)

Simple, Reliable, Accurate

- No special recompiles Use any build, any compiler
- Analyzes dynamically generated or linked code
- Inspects third party libraries where source is unavailable
- Productive user interface
- Command line for automated regression analysis



I		Problem	Sources	Modules	State
⊞ P1	☒	Mismatched allocat…	delete2.cpp; new.c…	tbb_debug.dll	New
⊞ P2	☒	Mismatched allocat…	find_and_fix_mem…	find_and_fix_me…	Confirmed
⊞ P3	☒	Kernel resource leak	asctime.c	MSVCR100D.dll	Deferred

Description	Source	Function	Module	Obje...	Off..
Mismatched deallocation site	find_and_fix_…	operator()	find_and_fix…		

```
                    drawing->put_pix
                }
                free(drawing);   //Me
                //delete drawing;
            }
```

```
find_and_fix_memory_errors.
find_and_fix_memory_errors.
tbb_debug.dll!local_wait_fo
tbb_debug.dll!local_spawn_r
tbb_debug.dll!spawn_root_an
```

Description	Source	Function	Module	Obje...	Off..
Allocation site	find_and_fix_…	operator()	find_and_fix…		

```
        for (int y = r.begin(); y !=
            {
                drawing_area * drawi
                for (int x = startx
                    color_t c = rend
```

```
find_and_fix_memory_errors.
find_and_fix_memory_errors.
tbb_debug.dll!local_wait_fo
tbb_debug.dll!local_spawn_r
tbb_debug.dll!spawn_root_an
```

Clicking an error instantly displays source code snippets and the call stack

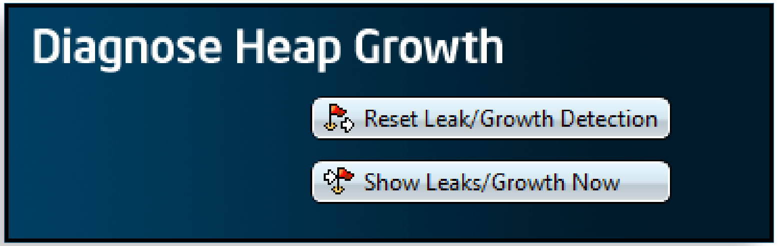
Easy to fit into your existing process

Optimization Notice

New for 2013! NEW

Intel® Inspector XE 2013 Dynamic Memory & Thread Analysis

Heap Growth Analysis



Diagnose heap growth. Get a list of memory allocations not freed in an interval set with the GUI or an API.

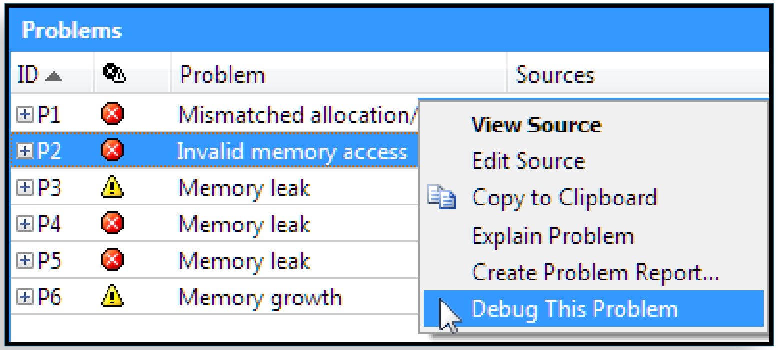
Improved Error Suppression

Precise Suppressions Remove False Errors Safely

```
Suppression = {
    Name = "Example";
    Type = { uninitialized_memory_access }
    Stacks = {
        {
            mod=a.out, func=update_x;
            func=main;
```

More precise, easy to edit, team shareable. Choose which stack frame to suppress. Eliminate the false, not the real errors.

Debugger Breakpoints



ID ▲		Problem	Sources
⊞ P1	⊗	Mismatched allocation/	
⊞ P2	⊗	Invalid memory access	
⊞ P3	⚠	Memory leak	
⊞ P4	⊗	Memory leak	
⊞ P5	⊗	Memory leak	
⊞ P6	⚠	Memory growth	

Diagnose the problem. Break into the debugger just before the error occurs. Examine the variables and threads.

Pause/Resume Collection

```
__itt_suppress_push(__itt_suppress_threading_errors);
    /* Any threading errors here are ignored */
__itt_suppress_pop();
    /* Any threading errors here are seen */
```

Speed-up analysis by limiting its scope. Turn on analysis only during the execution of the suspected problem.

Find and diagnose errors with less effort.

Optimization Notice

Analysis - Intel® Inspector XE

What's New in SP1?

Easier Migration From Other Tools

- Import suppression lists from Purify* and Valgrind* on Linux*

Fewer False Errors and Easier Suppression Management

- Precise suppressions specify single or multiple stack locations
- User editable suppression files (or use the GUI)
- Fortran – reduced false positives due to allocation

Leak Reports No Waiting!

- Set a baseline for incremental analysis with GUI or API
- Report incremental leaks and heap growth since the baseline
- No waiting until the end of the analysis run

New OS, Threading Model & Processor Support

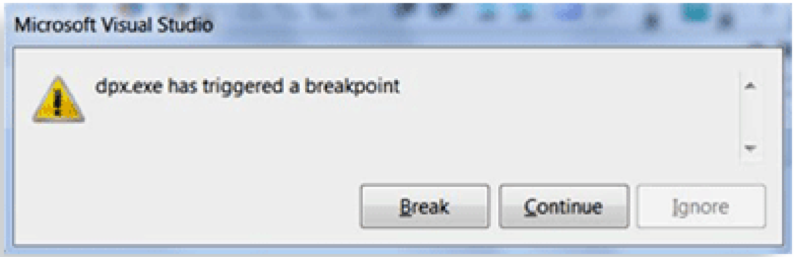
- OpenMP 4.0
- Haswell – Windows* & Linux*
- Windows* 8 desktop
- Visual Studio* 2012
- Latest Linux* distributions

New since the first 2013 release. Some features released in earlier updates.

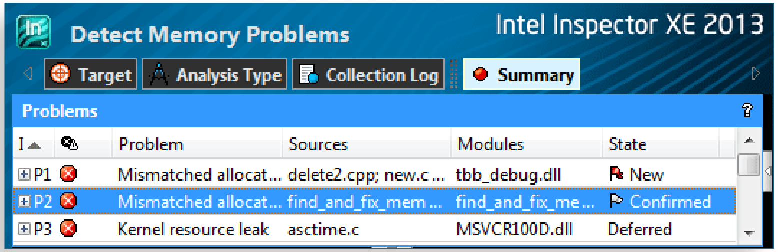
Pointer Checker and Memory Checker

Intel Parallel Studio XE family of suites

Pointer Checker NEW	Memory Checker
Recompile with Intel® Compiler	Use any build, any compiler
Lower overhead	Higher overhead
Only finds pointer errors	Finds multiple error types
One error at a time	GUI sorts multiple errors
Traceback: Source file + Line #	Traceback: Shows source code
Triggers debugger breakpoint	Triggers debugger breakpoint



dpx.exe has triggered a breakpoint



I		Problem	Sources	Modules	State
⊞ P1	⊗	Mismatched allocat...	delete2.cpp; new.c ...	tbb_debug.dll	New
⊞ P2	⊗	Mismatched allocat...	find_and_fix_mem ...	find_and_fix_me...	Confirmed
⊞ P3	⊗	Kernel resource leak	asctime.c	MSVCR100D.dll	Deferred

Two great ways to create more reliable software

Optimization Notice

Static Analysis Finds Coding and Security Errors

Intel® Parallel Studio XE 2013 Family of Suites

Find over 250 error types

- Incorrect directives, memory leaks, pointer and array errors, buffer overflows, uninitialized variables…

Easier to use NEW

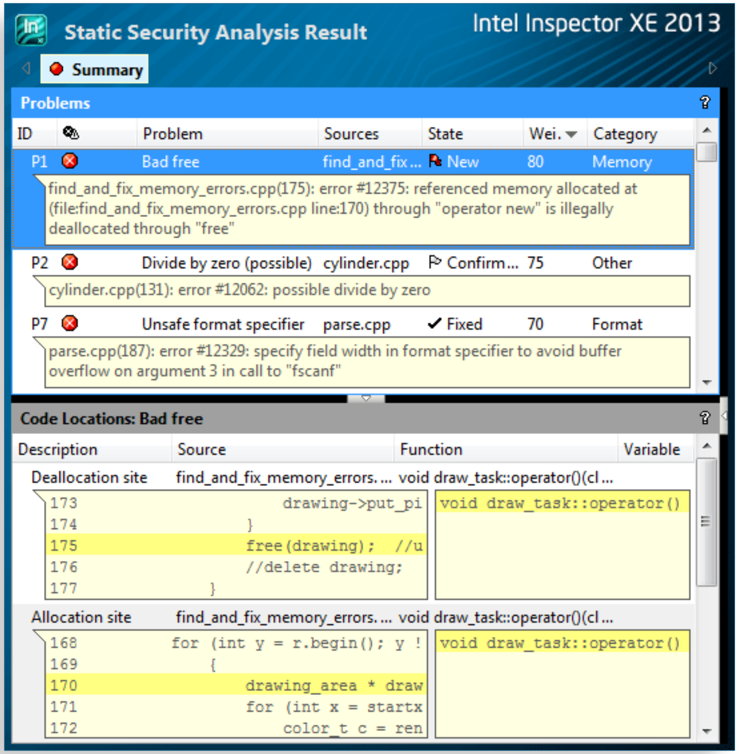
- Choose your priority:
  - Minimize false errors
  - Maximize error detection
- Hierarchical navigation of results
- Share comments with the team

Increased Accuracy & Speed NEW

- Detect errors without all source files
- Better scaling with large code bases

Code Complexity Metrics NEW

- Find code likely to be less reliable



Static Security Analysis Result

Intel Inspector XE 2013

● Summary

Problems

ID		Problem	Sources	State	Wei.	Category
P1	⊗	Bad free	find_and_fix…	New	80	Memory
		find_and_fix_memory_errors.cpp(175): error #12375: referenced memory allocated at (file:find_and_fix_memory_errors.cpp line:170) through "operator new" is illegally deallocated through "free"				
P2	⊗	Divide by zero (possible)	cylinder.cpp	Confirm…	75	Other
		cylinder.cpp(131): error #12062: possible divide by zero				
P7	⊗	Unsafe format specifier	parse.cpp	✓ Fixed	70	Format
		parse.cpp(187): error #12329: specify field width in format specifier to avoid buffer overflow on argument 3 in call to "fscanf"				

Code Locations: Bad free

Description	Source	Function	Variable
Deallocation site	find_and_fix_memory_errors.…	void draw_task::operator()(cl…	

```
173                    drawing->put_pi
174                }
175                free(drawing);  //u
176                //delete drawing;
177            }
```

```
void draw_task::operator()
```

Description	Source	Function	Variable
Allocation site	find_and_fix_memory_errors.…	void draw_task::operator()(cl…	

```
168        for (int y = r.begin(); y !
169            {
170                drawing_area * draw
171                for (int x = startx
172                    color_t c = ren
```

```
void draw_task::operator()
```

Clicking an error instantly displays source code snippets and traceback. Available for C, C++ and Fortran.

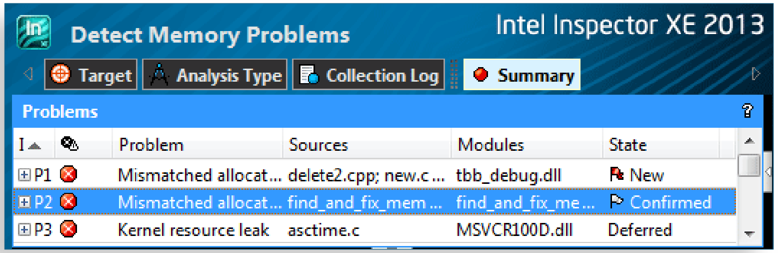
Find Errors and Harden your Security

Static Analysis is only available in the Parallel Studio XE family of suites. It is not sold separately.

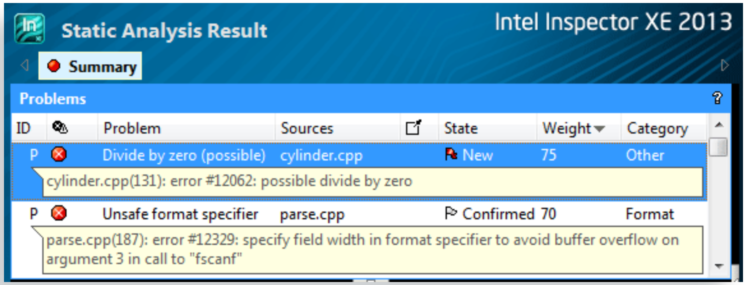
Dynamic Analysis Complements Static Analysis

In Intel® Parallel Studio XE family suites

Dynamic Analysis	Static Analysis
Use any build, any compiler	Rebuild with Intel® Compiler (Keep your existing compiler for code generation.)
Fewer false errors. Only active code paths are analyzed.	Comprehensive, but more false errors. Not limited by test cases.
Analyze 3rd party code	n/a – Source required
Can trigger debugger breakpoint	n/a – No diagnostic capability
Slow (1x – 20x - 100x workload)	Fast (no workload, “slow” build)
Memory & Threading Errors	Memory, Code & Security Errors



I		Problem	Sources	Modules	State
⊞ P1	⊗	Mismatched allocat…	delete2.cpp; new.c…	tbb_debug.dll	New
⊞ P2	⊗	Mismatched allocat…	find_and_fix_mem…	find_and_fix_me…	Confirmed
⊞ P3	⊗	Kernel resource leak	asctime.c	MSVCR100D.dll	Deferred



ID		Problem	Sources		State	Weight	Category
P	⊗	Divide by zero (possible)	cylinder.cpp		New	75	Other
P	⊗	Unsafe format specifier	parse.cpp		Confirmed	70	Format

cylinder.cpp(131): error #12062: possible divide by zero

parse.cpp(187): error #12329: specify field width in format specifier to avoid buffer overflow on argument 3 in call to "fscanf"

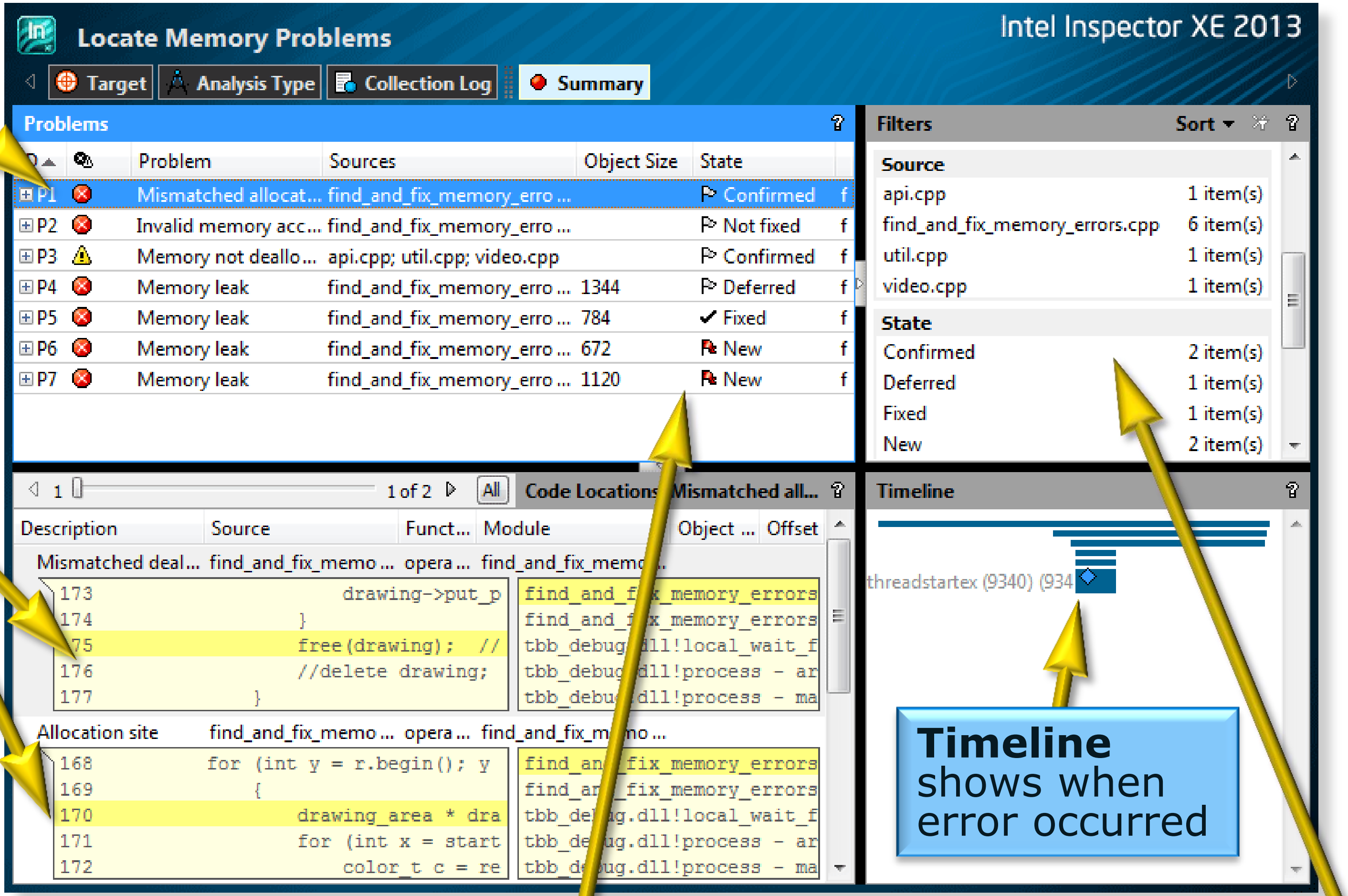
Two great ways to create more reliable software

Productive User Interface

Intel® Inspector XE

Select a **problem set**

Code snippets displayed for selected problem



Problems

ID		Problem	Sources	Object Size	State	
P1	⊗	Mismatched allocat...	find_and_fix_memory_erro...		Confirmed	f
P2	⊗	Invalid memory acc...	find_and_fix_memory_erro...		Not fixed	f
P3	⚠	Memory not deallo...	api.cpp; util.cpp; video.cpp		Confirmed	f
P4	⊗	Memory leak	find_and_fix_memory_erro...	1344	Deferred	f
P5	⊗	Memory leak	find_and_fix_memory_erro...	784	✓ Fixed	f
P6	⊗	Memory leak	find_and_fix_memory_erro...	672	New	f
P7	⊗	Memory leak	find_and_fix_memory_erro...	1120	New	f

Filters Sort

Source	
api.cpp	1 item(s)
find_and_fix_memory_errors.cpp	6 item(s)
util.cpp	1 item(s)
video.cpp	1 item(s)

State	
Confirmed	2 item(s)
Deferred	1 item(s)
Fixed	1 item(s)
New	2 item(s)

1 1 of 2 All Code Locations: Mismatched all...

Description	Source	Funct...	Module	Object ...	Offset
Mismatched deal...	find_and_fix_memo...	opera...	find_and_fix_memo...		

```
                        drawing->put_p
                    }
                    free(drawing);  //
                    //delete drawing;
               }
```

```
find_and_fix_memory_errors
find_and_fix_memory_errors
tbb_debug.dll!local_wait_f
tbb_debug.dll!process - ar
tbb_debug.dll!process - ma
```

Description	Source	Funct...	Module	Object ...	Offset
Allocation site	find_and_fix_memo...	opera...	find_and_fix_memo...		

```
          for (int y = r.begin(); y
               {
                    drawing_area * dra
                    for (int x = start
                         color_t c = re
```

```
find_and_fix_memory_errors
find_and_fix_memory_errors
tbb_debug.dll!local_wait_f
tbb_debug.dll!process - ar
tbb_debug.dll!process - ma
```

Timeline

Timeline shows when error occurred

Problem States: New, Not Fixed, Fixed, Confirmed, Not a problem, Regression

Filters let you focus on a module, or error type, or…

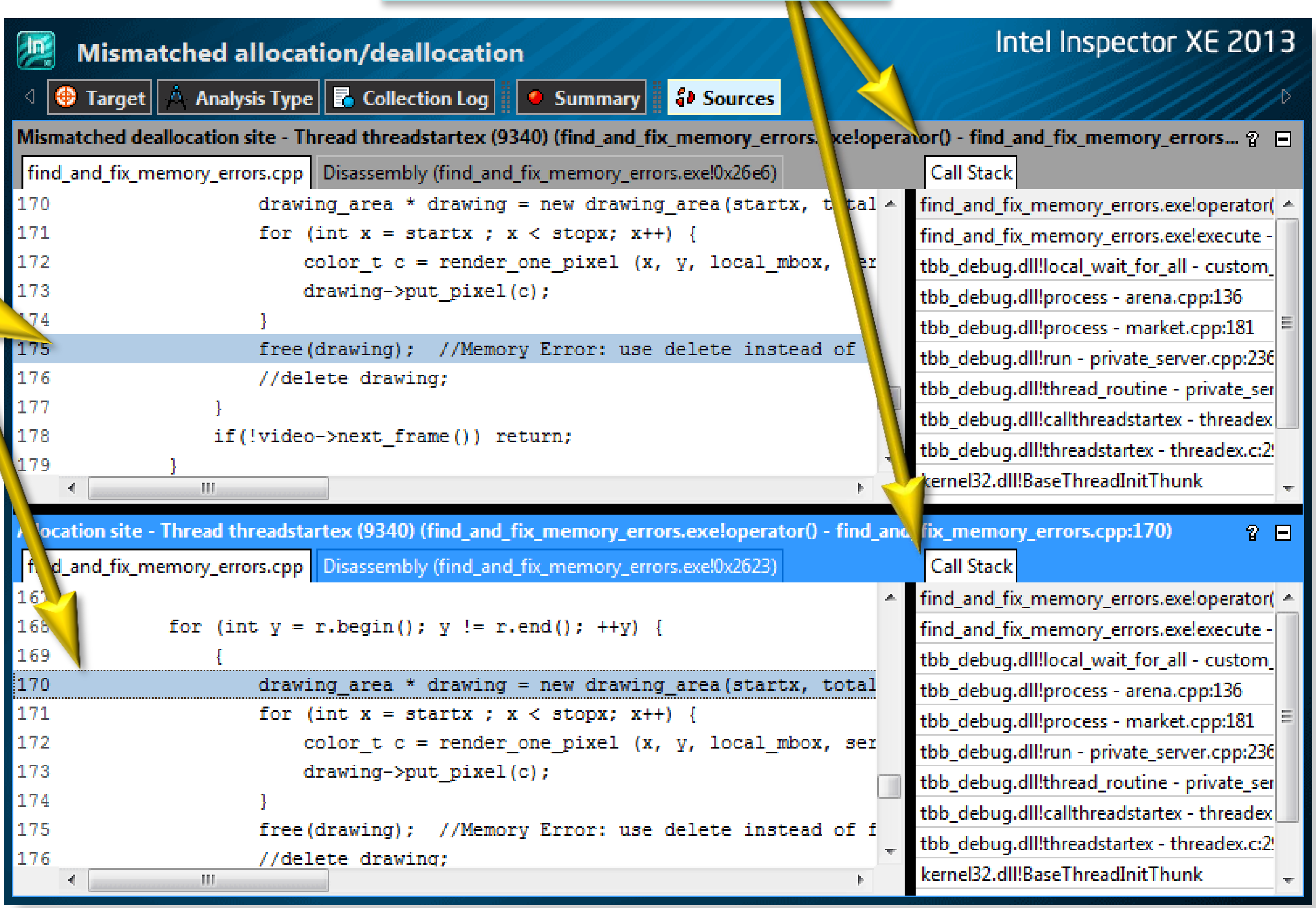
Optimization Notice

Double Click for Source & Call Stack

Intel® Inspector XE

Call Stack

Source code locations displayed for selected problem



Mismatched allocation/deallocation
Intel Inspector XE 2013
Target
Analysis Type
Collection Log
Summary
Sources

Mismatched deallocation site - Thread threadstartex (9340) (find_and_fix_memory_errors.exe!operator() - find_and_fix_memory_errors...

find_and_fix_memory_errors.cpp
Disassembly (find_and_fix_memory_errors.exe!0x26e6)
Call Stack

```
            drawing_area * drawing = new drawing_area(startx, total
            for (int x = startx ; x < stopx; x++) {
                color_t c = render_one_pixel (x, y, local_mbox, ser
                drawing->put_pixel(c);
            }
            free(drawing);  //Memory Error: use delete instead of
            //delete drawing;
        }
        if(!video->next_frame()) return;
    }
```

```
find_and_fix_memory_errors.exe!operator(
find_and_fix_memory_errors.exe!execute -
tbb_debug.dll!local_wait_for_all - custom_
tbb_debug.dll!process - arena.cpp:136
tbb_debug.dll!process - market.cpp:181
tbb_debug.dll!run - private_server.cpp:23
tbb_debug.dll!thread_routine - private_ser
tbb_debug.dll!callthreadstartex - threadex
tbb_debug.dll!threadstartex - threadex.c:2
kernel32.dll!BaseThreadInitThunk
```

Allocation site - Thread threadstartex (9340) (find_and_fix_memory_errors.exe!operator() - find_and_fix_memory_errors.cpp:170)

find_and_fix_memory_errors.cpp
Disassembly (find_and_fix_memory_errors.exe!0x2623)
Call Stack

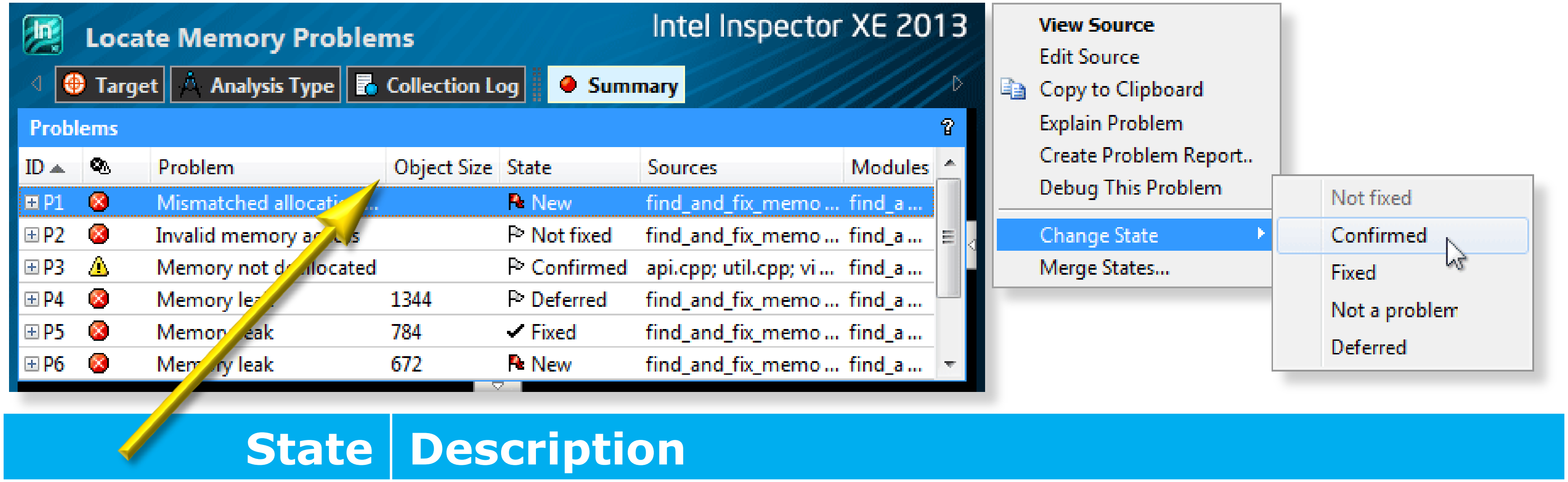
```

    for (int y = r.begin(); y != r.end(); ++y) {
        {
            drawing_area * drawing = new drawing_area(startx, total
            for (int x = startx ; x < stopx; x++) {
                color_t c = render_one_pixel (x, y, local_mbox, ser
                drawing->put_pixel(c);
            }
            free(drawing);  //Memory Error: use delete instead of f
            //delete drawing;
```

```
find_and_fix_memory_errors.exe!operator(
find_and_fix_memory_errors.exe!execute -
tbb_debug.dll!local_wait_for_all - custom_
tbb_debug.dll!process - arena.cpp:136
tbb_debug.dll!process - market.cpp:181
tbb_debug.dll!run - private_server.cpp:23
tbb_debug.dll!thread_routine - private_ser
tbb_debug.dll!callthreadstartex - threadex
tbb_debug.dll!threadstartex - threadex.c:2
kernel32.dll!BaseThreadInitThunk
```

Optimization Notice

Problem State Lifecycle

Makes problems easier to manage



ID		Problem	Object Size	State	Sources	Modules
P1	⊗	Mismatched allocati...		New	find_and_fix_memo...	find_a...
P2	⊗	Invalid memory access		Not fixed	find_and_fix_memo...	find_a...
P3	⚠	Memory not deallocated		Confirmed	api.cpp; util.cpp; vi...	find_a...
P4	⊗	Memory leak	1344	Deferred	find_and_fix_memo...	find_a...
P5	⊗	Memory leak	784	✓ Fixed	find_and_fix_memo...	find_a...
P6	⊗	Memory leak	672	New	find_and_fix_memo...	find_a...

State	Description
New	Detected by this run
Not Fixed	Previously seen error detected by this run
Not a Problem	Set by user (tool will not change)
Confirmed	Set by user (tool will not change)
Fixed	Set by user (tool will change)
Regression	Error detected with previous state of "Fixed"

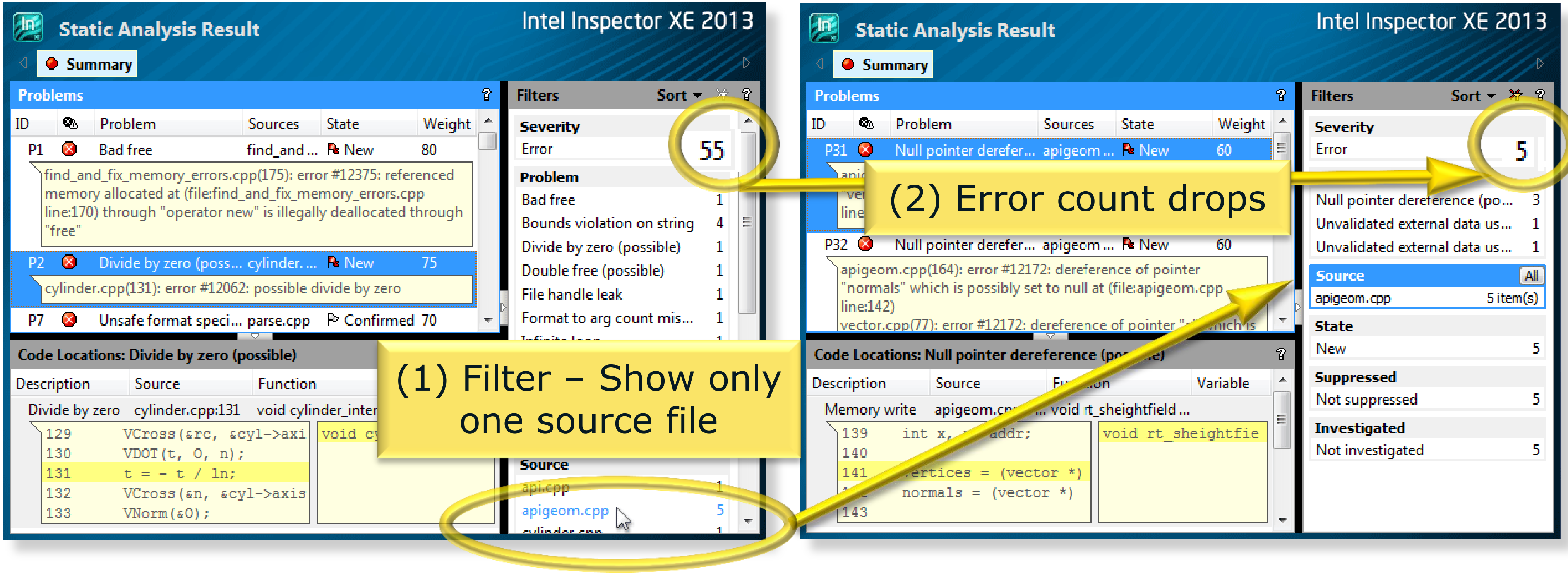
Optimization Notice

Filtering - Focus on what’s important

Example: See only the errors in one source file

Before – All Errors

After – Only errors from one source file



ID		Problem	Sources	State	Weight
P1	☒	Bad free	find_and ...	New	80
P2	☒	Divide by zero (poss...	cylinder. ...	New	75
P7	☒	Unsafe format speci...	parse.cpp	Confirmed	70

find_and_fix_memory_errors.cpp(175): error #12375: referenced memory allocated at (file:find_and_fix_memory_errors.cpp line:170) through "operator new" is illegally deallocated through "free"

cylinder.cpp(131): error #12062: possible divide by zero

Filters	
Severity	
Error	55
Problem	
Bad free	1
Bounds violation on string	4
Divide by zero (possible)	1
Double free (possible)	1
File handle leak	1
Format to arg count mis...	1
Infinite loop	1
Source	
api.cpp	1
apigeom.cpp	5
cylinder.cpp	1

Code Locations: Divide by zero (possible)

Description	Source	Function
Divide by zero	cylinder.cpp:131	void cylinder_inter

```
129        VCross(&rc, &cyl->axi
130        VDOT(t, O, n);
131        t = - t / ln;
132        VCross(&n, &cyl->axis
133        VNorm(&O);
```

```
void cy
```

(1) Filter – Show only one source file

(2) Error count drops

ID		Problem	Sources	State	Weight
P31	☒	Null pointer derefer...	apigeom ...	New	60
P32	☒	Null pointer derefer...	apigeom ...	New	60

apigeom.cpp(164): error #12172: dereference of pointer "normals" which is possibly set to null at (file:apigeom.cpp line:142)
vector.cpp(77): error #12172: dereference of pointer " which is

Filters	
Severity	
Error	5
Null pointer dereference (po...	3
Unvalidated external data us...	1
Unvalidated external data us...	1
Source	All
apigeom.cpp	5 item(s)
State	
New	5
Suppressed	
Not suppressed	5
Investigated	
Not investigated	5

Code Locations: Null pointer dereference (po e)

Description	Source	Function	Variable
Memory write	apigeom.c ...	void rt_sheightfield ...	

```
139     int x,    addr;
140
141     ertices = (vector *)
        normals = (vector *)
143
```

```
void rt_sheightfie
```

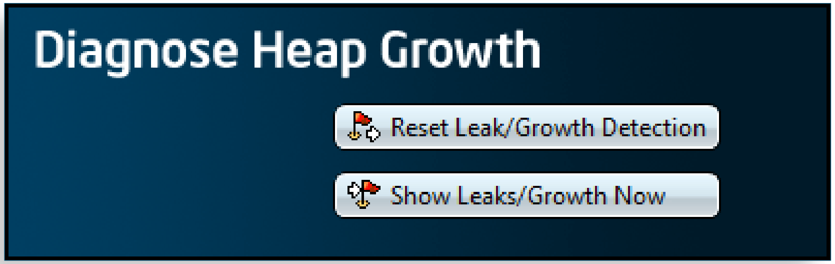
Tip: Set the “Investigated” filter to “Not investigated” while investigating problems. This removes from view the problems you are done with, leaving only the ones left to investigate.

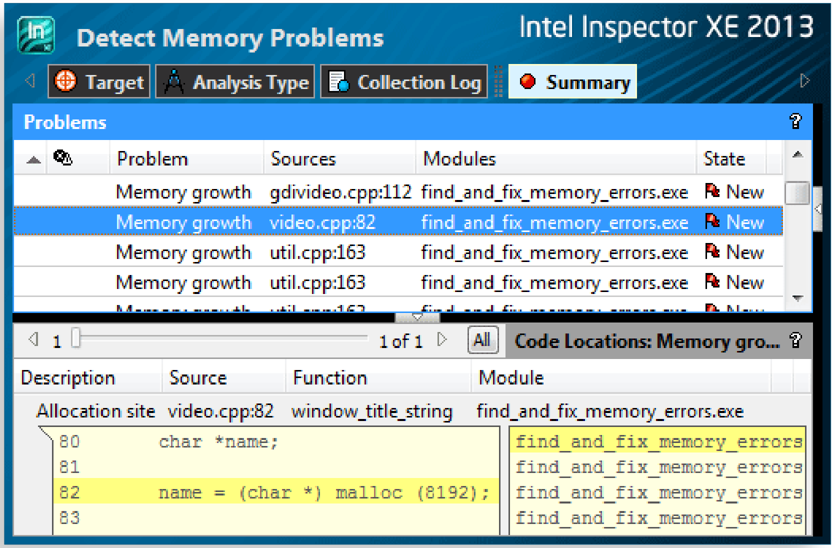
Static Analysis shown, but filters work the same way for dynamic memory & threading analysis.

Heap Growth Analysis NEW

Does your memory usage grow mysteriously?

- Set an analysis interval with start and analysis end points
  - Click a button –or–
  - Use an API
- See a list of memory allocations that are not freed in the interval
- Quickly zero in on suspicious activity that contributes to heap growth





Problem	Sources	Modules	State
Memory growth	gdivideo.cpp:112	find_and_fix_memory_errors.exe	New
Memory growth	video.cpp:82	find_and_fix_memory_errors.exe	New
Memory growth	util.cpp:163	find_and_fix_memory_errors.exe	New
Memory growth	util.cpp:163	find_and_fix_memory_errors.exe	New

Description	Source	Function	Module
Allocation site	video.cpp:82	window_title_string	find_and_fix_memory_errors.exe

```
80      char *name;
81
82      name = (char *) malloc (8192);
83
```

```
find_and_fix_memory_errors
find_and_fix_memory_errors
find_and_fix_memory_errors
find_and_fix_memory_errors
```

Speed diagnosis of difficult to find heap errors

Command Line Interface

Automate analysis

inspxe-cl is the command line:

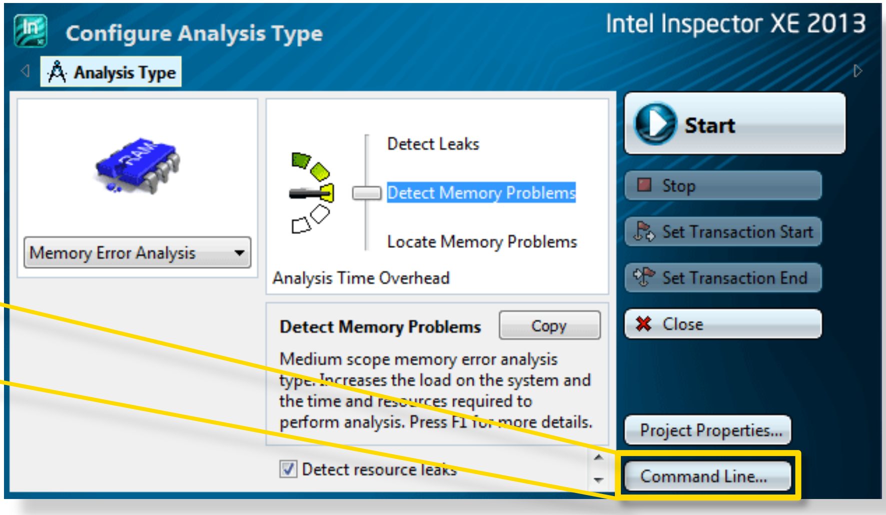
- Windows: `C:\Program Files\Intel\Inspector XE \bin[32|64]\inspxe-cl.exe`
- Linux: `/opt/intel/inspector_xe/bin[32|64]/inspxe-cl`

Help:

```
inspxe-cl -help
```

Set up command line with GUI

Command Line...



Detect Memory Problems

Medium scope memory error analysis type. Increases the load on the system and the time and resources required to perform analysis. Press F1 for more details.

☑ Detect resource leaks

Command examples:

```
1. inspxe-cl -collect-list
2. inspxe-cl -collect ti2 -- MyApp.exe
3. inspxe-cl -report problems
```

Great for regression analysis – send results file to developer
Command line results can also be opened in the GUI

Optimization Notice

Productive User Interface

Intel® Inspector XE

	Dynamic	Static
View Context of Problem		
Stack	✓	✓
Multiple Contributing Source Locations	✓	✓
Collapse multiple “sightings” to one error (e.g., memory allocated in a loop, then leaked is 1 error)	✓	✓
Suppression, Filtering, and Workflow Management	✓	✓
Visual Studio* Integration (Windows*)	✓	✓
Command line for automated tests	✓	✓
Time Line visualization	✓	
Memory Growth during a transaction	✓	
Trigger Debugger Breakpoint	✓	

One productive interface for both static and dynamic analysis.

Static Analysis is included in Parallel Studio XE studio bundles. It is not sold separately.

Intel® Parallel Studio XE Suites

Leading development suite for application performance

	Intel® Cluster Studio XE	Intel® Parallel Studio XE	
Analysis	●	●	**Intel® VTune™ Amplifier XE** - Performance Profiler
	●	●	**Intel® Inspector XE** - Memory & Thread Analyzer
	●	●	**Static Analysis & Pointer Checker** - Find Coding & Security Errors
	●	●	**Intel® Advisor XE** - Threading Assistant
	●		**Intel® Trace Analyzer & Collector** - MPI Optimizing Tool
Compilers & Libraries	●	●	**Intel® Compiler** - Optimizing Compiler for C, C++ and Fortran
	●	●	**Intel® Integrated Performance Primitives**† - Media and Data Optimizations
	●	●	**Intel® Threading Building Blocks**† **-** Parallelize Applications for Performance
	●	●	**Intel® Math Kernel Library -** High Performance Math
	●		**Intel® MPI Library** - Flexible, Efficient and Scalable Messaging

† Available for C, C++ only

C, C++ only and Fortran only versions of Parallel Studio XE are also available.

Create fast, reliable code

Optimization Notice

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Backup

Dynamic Analysis Finds Hidden Errors Early

Intel® Inspector XE 2013

Cross-thread Stack Access

Occurs when a thread accesses a different thread's stack.

Data Race

Occurs when multiple threads access the same memory location without proper synchronization and at least one access is a write.

Deadlock

Occurs when two or more threads are waiting for each other to release resources (such as mutexes, critical sections, and thread handles) while holding resources the other threads are trying to acquire. If none of the threads release their resources, then none of the threads can proceed.

GDI Resource Leak

Occurs when a GDI object is created but never deleted.

Incorrect memcpy Call

Occurs when an application calls the memcpy function with two pointers that overlap within the range to be copied. This condition is only checked on Linux* systems. On Windows* systems, this function is safe for overlapping memory.

Invalid Deallocation

Occurs when an application calls a deallocation function with an address that does not correspond to dynamically allocated memory.

Invalid Memory Access

Occurs when a read or write instruction references memory that is logically or physically invalid.

Invalid Partial Memory Access

Occurs when a read or write instruction references a block (2-bytes or more) of memory where part of the block is logically invalid.

Kernel Resource Leak

Occurs when a kernel object handle is created but never closed.

Lock Hierarchy Violation

Occurs when the acquisition order of multiple synchronization objects (such as mutexes, critical sections, and thread handles) in one thread differs from the acquisition order in another thread, and these synchronization objects are owned by the acquiring thread and must be released by the same thread.

Memory Growth

Occurs when a block of memory is allocated but not deallocated within a specific time segment during application execution.

Memory Leak

Occurs when a block of memory is allocated and never released.

Mismatched Allocation/Deallocation

Occurs when a deallocation is attempted with a function that is not the logical reflection of the allocator used.

Missing Allocation

Occurs when an invalid pointer is passed to a deallocation function. The invalid address may point to a previously released heap block.

Thread Start Information

Occurs when the Intel Inspector XE detects the creation of a thread. This *problem* is really informational feedback useful for confirming the number and location of threads created during application execution and data collection.

Unhandled Application Exception

Occurs when the application undergoing analysis crashes because of an unhandled exception thrown by the application.

Uninitialized Memory Access

Occurs when a read of an uninitialized memory location is reported.

Uninitialized Partial Memory Access

Occurs when a read instruction references a block (2-bytes or more) of memory where part of the block is uninitialized.

For details, see our online documentation.

Optimization Notice

Static Analysis Finds Over 250 Kinds of Errors

Intel® Parallel Studio XE 2013 family of suites

Here are some examples…

- ALLOCATABLE array referenced before allocation
- Argument corresponding to * for width or precision value should be type int
- Argument count mismatch
- Argument count mismatch at call to intrinsic function
- Argument is not a pointer
- Argument type mismatch at call to intrinsic function
- Array parameter element size mismatch
- Array parameter rank mismatch
- Array parameter shape mismatch
- Attempt to violate exception specification
- Bad format flags
- Base class has non-virtual destructor
- Base class lacks destructor
- Big parameter passed by value
- Bounds violation
- Buffer overflow through pointer
- C library routine violates C++ object semantics
- Chunk_size in OpenMP* SCHEDULE clause has side-effects
- Chunk_size in OpenMP* SCHEDULE clause not loop-invariant
- Class has virtual member functions but no derived classes
- COMMON block is partly OpenMP* THREADPRIVATE
- Conditional OpenMP* BARRIER
- Data race
- Data race from cilk_for
- Data race from cilk_spawn
- Destructor contains non-empty exception specification
- Divide by zero
- Double free
- Duplicate subroutine definition
- Exception thrown from destructor
- File closed twice
- Format to argument count mismatch
- Format to argument type mismatch
- FORTRAN IN argument modified
- Function illegally exits OpenMP* construct
- Function result ignored
- Function result not set
- Function return value discarded
- Function use does not match its definition
- Gets function is unsafe
- Global object constructor can throw exception
- Global object destructor can throw exception
- Global redefinition of new or delete
- Global/static variable relies on default initialization
- Illegal parameter value
- Implicit function declaration
- Implicit type conversion causes object slicing
- Improper nesting of OpenMP* constructs
- Improper nesting of OpenMP* CRITICAL directives
- Improper use of intrinsic function
- Improper use of OpenMP* PRIVATE variable
- Improper use of OpenMP* REDUCTION variable
- Improper use of OpenMP* THREADPRIVATE array
- Improper use of OpenMP* THREADPRIVATE variable
- Inconsistent array declaration (element count mismatch)
- Inconsistent array declaration (element size mismatch)
- Inconsistent array declaration (element type mismatch)
- Inconsistent array declaration (size mismatch)
- Inconsistent enumeration declaration (enum value mismatch)
- Inconsistent enumeration declaration (member count mismatch)
- Inconsistent enumeration declaration (name mismatch)
- Inconsistent enumeration declaration (tag mismatch)
- Inconsistent enumeration declaration (type mismatch)
- Inconsistent pointer declaration (size mismatch)
- Inconsistent pointer declaration (target size mismatch)
- Inconsistent pointer declaration (type mismatch)
- Inconsistent string declaration
- Inconsistent structure declaration (field offset mismatch)
- Inconsistent structure/union declaration (field count mismatch)
- Inconsistent structure/union declaration (field name mismatch)
- Inconsistent structure/union declaration (field size mismatch)
- Inconsistent structure/union declaration (field type mismatch)
- Inconsistent structure/union declaration (size mismatch)
- Inconsistent structure/union declaration (tag mismatch)
- Inconsistent structure/union declaration (type mismatch)

For a more complete list, see our online documentation.