Debugging Errors with Windows WinDbg

Quote from Microsoft Development Lead at Microsoft

"The latest releases of Microsoft Windows use the same operating system kernel, the same primary interfaces, drivers work on both server and client, and the debugger uses the same debug files. Furthermore, we used the same code base and source tree to compile both 32- and 64-bit versions."

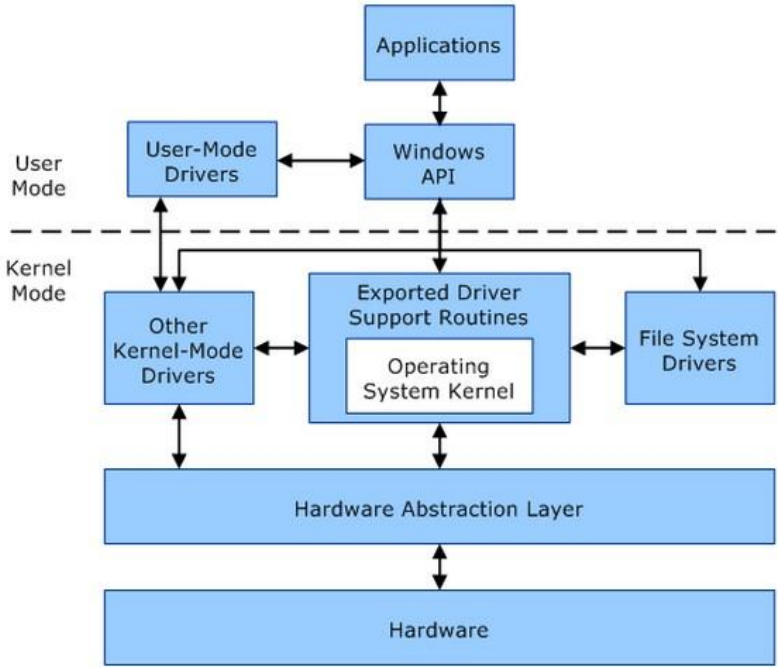
Therefore, the techniques described here will work on all Windows Platforms.
Now that we understand that let's continue with some basic understanding of the similarities

User mode and kernel mode basic definitions from Microsoft

A processor in a computer running Windows has two different modes: user mode and kernel mode. The processor switches between the two modes depending on what type of code is running on the processor. Applications run in user mode, and core operating system components run in kernel mode. Many drivers run in kernel mode, but some drivers run in user mode.

When you start a user-mode application, Windows creates a process for the application. The process provides the application
with a private virtual address space and a private handle table. Because an application's virtual address space is private, one application cannot alter data that belongs to another application. Each application runs in isolation, and if an application crashes, the crash is limited to that one application. Other applications and the operating system are not affected by the crash.
In addition to being private, the virtual address space of a user-mode application is limited. A processor running in user mode
cannot access virtual addresses that are reserved for the operating system. Limiting the virtual address space of a user-mode
application prevents the application from altering, and possibly damaging, critical operating system data.

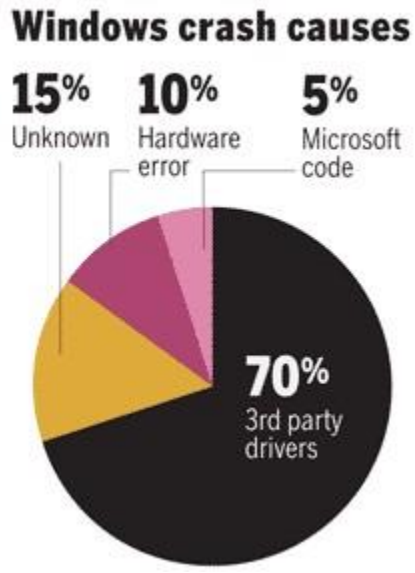
All code that runs in kernel mode shares a single virtual address space. This means that a kernel-mode driver is not isolated from other drivers and the operating system itself. If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs to the operating system or another driver could be compromised. If a kernel-mode driver crashes, the entire
operating system crashes.



Note: The Windows API, informally WinAPI, is Microsoft's core set of application programming interfaces (APIs) available in the
Microsoft Windows operating systems. Almost all Windows programs interact with the Windows API.
Almost every new version of Microsoft Windows has introduced its own additions and changes to the Windows API.
User Mode software cannot directly access the hardware or reference any address freely. It must pass instructions/requests through
calls to API. Crashes in User Mode are generally recoverable, requiring a restart of the application, but not the entire system.

But kernel-mode software is not protected from other kernel-mode software. For example, if a video driver erroneously accesses a
portion of memory assigned to another program (or memory not marked as accessible to drivers) Windows will stop the entire system.
This is known as a Bug Check and the familiar Blue Screen of Death is displayed.

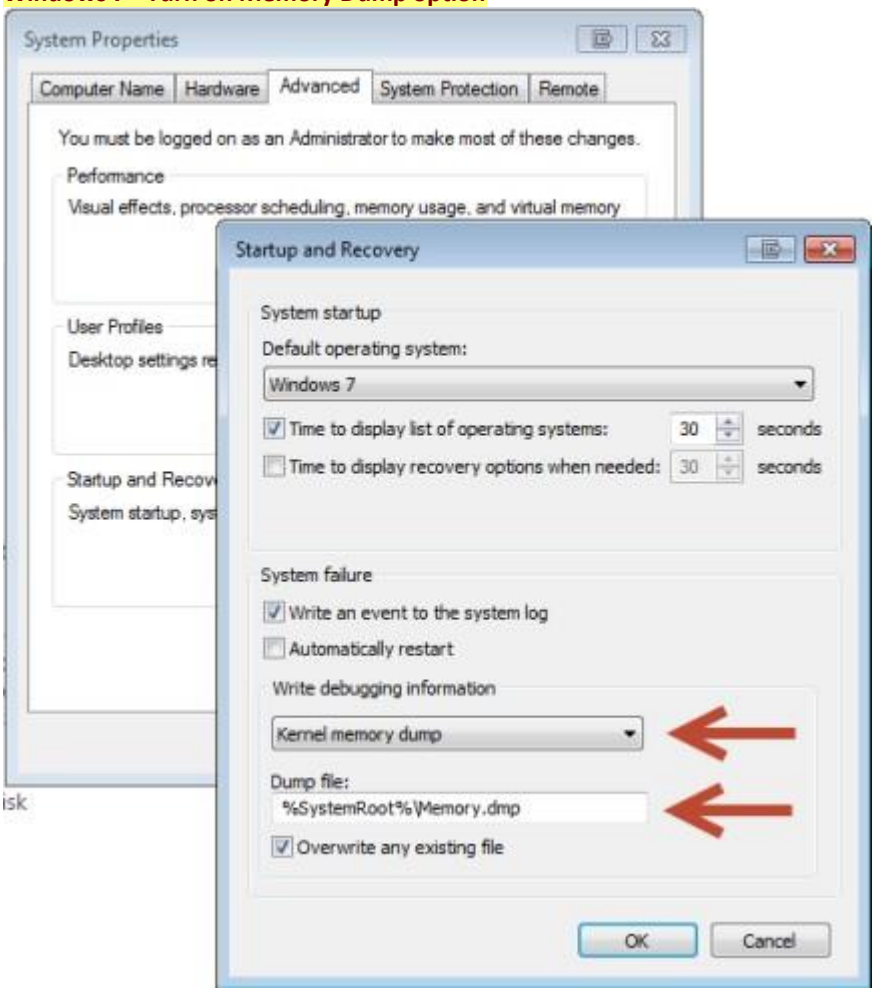
An analysis of Window's crashes yields the following statistics:



Now that we understand the basic reasons behind these crashes let's get started.
What you will need to DeBug Windows crashes

1> The latest version of WinDbg - It comes as part of the Windows SDK - Software Development Kit
2> A memory dump file to analyze.
3> Option to create a memory dump file turned on.

Windows 7 - Turn on Memory Dump option



Windows 8 - - Turn on Memory Dump option
NEW: Automatic Memory Dump Settings on Windows 8

4> Need to download the Microsoft DeBug Symbols. - Turn off Firewall!
These Symbols must also match the version of OS you are using. Symbol tables are a byproduct of the compilation of code. When a program is compiled, the source code is translated from a high-level language into machine code. At the same time, the compiler creates a symbol file as an identifier for the code. Helps to reduce the size of the exe.

Therefore, When a program causes a problem, the operating system knows only the hex address at which the problem occurred. You need something more than that to determine which program was using that memory space and what it was trying to do. Having the Symbols file can identify which program was trying to address a certain address space.

5> Configure WinDbg to locate the symbols on your system
launch WinDbg and select the following: File | Symbol file path
then enter the following command: srv*c:\symbols*http://msdl.microsoft.com/download/symbols
Note = **c:\symbols** in the above command is where you want to store the symbols

Also note that the symbols downloaded will only match the OS you are downloading them from. You can opt to download all the Symbol files for every OS if you plan to diagnose many different dump systems files from your one machine.

What is a Dump File?

A memory dump file is a snapshot of what the system had in memory when it crashed.
Windows creates three different sizes of memory dumps;

minidumps - They do not contain any of the binary or executable files that were in memory at the time of the failure. But they are not needed if the dump was created on the same machine you are analyzing it from.

kernel dumps - Kernel dumps are roughly equal in size to the RAM occupied by the Window's kernel. kernel dumps contains the binaries!

full dumps - A full memory dump is about equal to the amount of installed RAM.

Note:

practice dump files - files can be generated by a tool called "NotMyFault "

Generating a practice dump file

1> Download the program -"NotMyFault " or another program called StartBlueScreen if this doesn't work.
2> Run as Administrator
3> Select "High IRQL fault (kernelmode)" and the Do Bug button. Generates "Stop D1" error
4> Launch WinDbg as administrator
5> Select the menu option File | Open crash dump and point it to open the memory dump you want to analyze.
6> A Command window will appear. This is where the crash analysis will be displayed.
At the lower left will be a KD> prompt. To the right of the prompt is a single-line window where you will enter commands.
7> If you see this message, ***** Kernel symbols are WRONG. Please fix symbols to do analysis.
then WinDbg was unable to retrieve the proper symbols and it will resort to using the default symbol table.
This will not yield accurate results. Each version of Windows requires the collection of Symbols.
8> If successful, we can now start to analyze the crash.

Analyzing the Crash

Note: Microsoft does not store all of the third-party driver's Symbols and you could see errors indicating this.
Example: Unable to load image \??\C:\Windows\system32\drivers\myfault.sys, Win32 error 0n2
*** WARNING: Unable to verify timestamp for myfault.sys
*** ERROR: Module load completed but symbols could not be loaded for myfault.sys
Ignore these!!! You can still determine the issue/fault without these Symbols

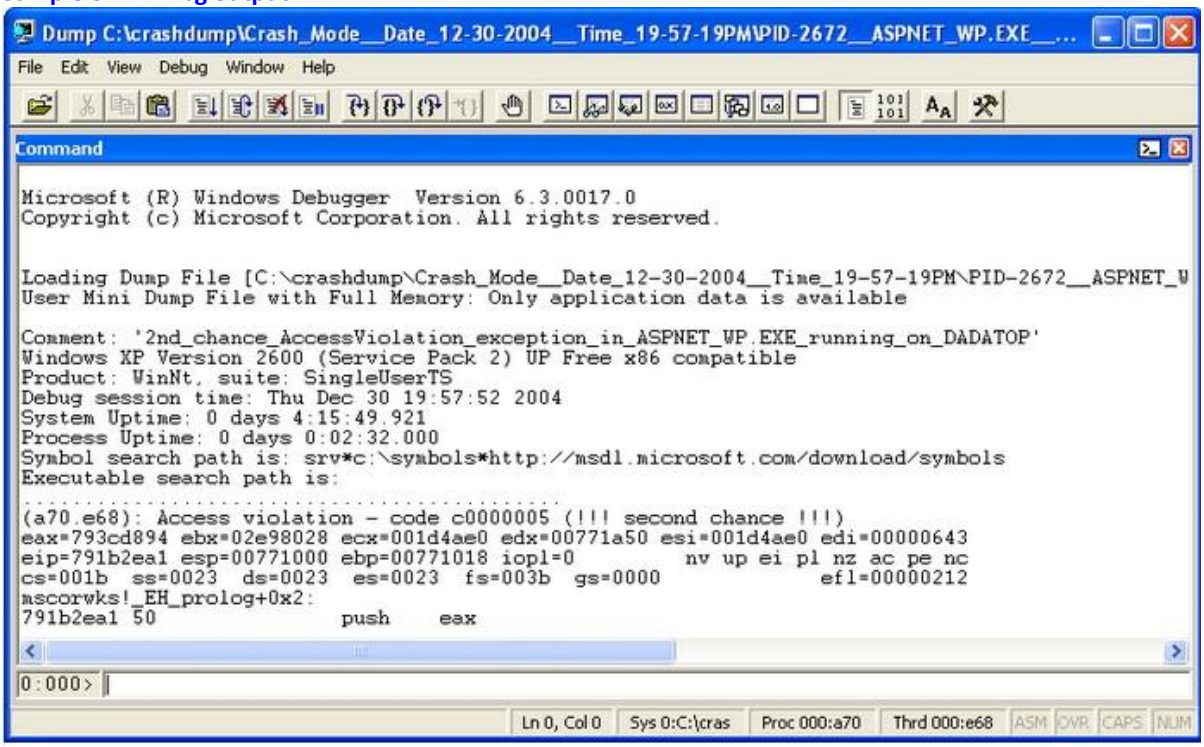
1> When WinDbg initially starts it automatically runs a basic analysis on the memory dump file.
Read through the file and you might even see a very obvious fault.
2> We only need a few commands to really troubleshoot the memory dump file: Look at the pic below:

!analyze -v	Analyze in Verbose Mode	!analyze –v displays information describing the state of a system when it crashed, the fault encountered, and who is the primary suspect.
lmv	Loaded Module Verbose	lmv displays a list of drivers and their path, version and vendor information. It often includes a product description. Output from lmv can take a long time. Watch the bottom left of the WnDbg interface where you would normally expect to see the kd> prompt. When retrieving information it will display *BUSY*. Only when the kd> prompt returns can you use additional commands.
lmvm [Module Name]	Loaded Module Verbose Module Name	lmvm [module name] enables you to tell the debugger to retrieve information for only that specific module. For example: lmvm myfault.sys

Type **!analyze -v** on the command line at the bottom of the Command window. -v = verbose mode

Note that the chronologic sequence of events goes from the bottom to the top; as each new task is performed by the system it shows up at the top, pushing the previous actions down.

Sample of WinDbg output



Note
If you see a "BugCheck Analysis" that is when the system stopped (Blue Screened).

3> Typing **lmv** in the command line displays the loaded modules; v instructs the debugger to output in verbose (detail) mode, showing all known details for the modules.

4> Locating the driver of interest can take a while, you can simplify the process by selecting **Edit | Find** and enter the suspected driver manufacturer name if known. Might get lucky

5> Once you know the name of the offending module you can isolate that information by typing in the following:

lmvm {name of module}

6> From here you can find the vendor's name. Now you can look for updates to the product or fixes mentioned on the site.

Note
Sometimes it is not so clear as to which driver is failing or whether or not their is a driver fault at all.
Ssometimes the information it provides is misleading or insufficient. In this you should start checking hardware.

First to check would be memory. You can use the follwoing tools:
Memtest86 / 86+ <4Gig
Memory Diags USB <4Gig
MS Memory Diags >4Gig
MS Win Mem Diags >4Gig

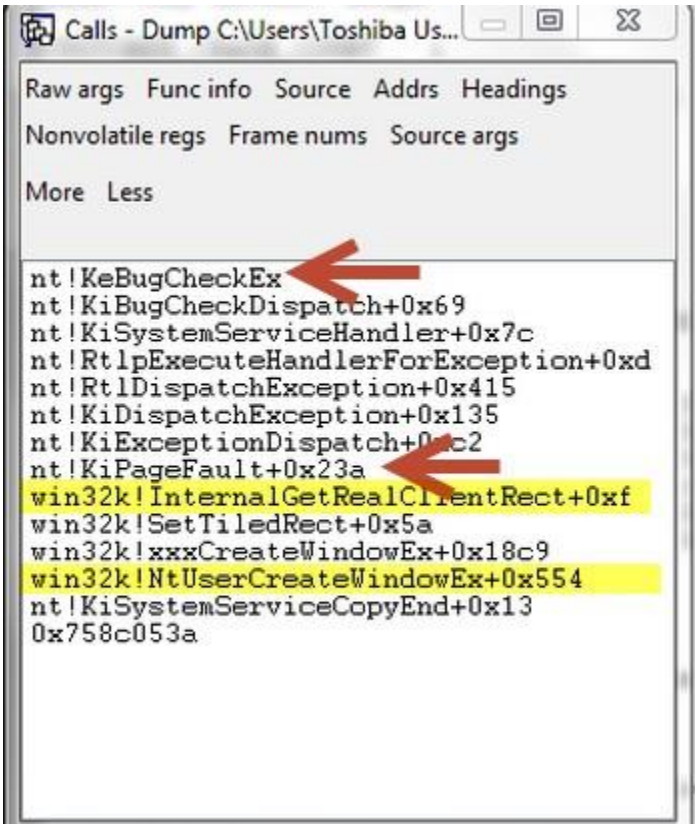
Note
You may see an Antivirus driver named as the culprit in some dumps, but be careful to quickly condem these programs. For antivirus code to work it must watch all file openings and closings. To accomplish this, the code sits at a low layer in the operating system and is constantly working. So make sure to look every where before claiming the AV is bad.

Note
Some driver vendors don't take the time to include sufficient information with their modules. So if lmv doesn't help, try looking at the subdirectories on the image path (if there is one).

Note
I have personally found that the BitBucket Error usually is my culprit when scanning dumps. More often than not it was so.

Note
For More Information You Can Check the Call Stack Window
The call stack window shows the last few calls that the thread made before the crash. It's available even with a Minidump. In fact, it's one of the few advanced windows that are available in a Minidump. Of course, the symbols play a huge role in deciphering what is going on in the call stack, too.

Sample Call Stack



The stack is read from the bottom to the top.
bugcheck - **nt!KeBugCheckEx** means "Start the Blue Screen of Death process." BSOD
It's the last thing you see before the blue screen, and the system starts making the memory dump that you're now viewing.
Reading back through the line, we can see where the thread stopped, and then the blue screen was set in motion:

win32k!InternalGetRealClientRect+0xf.

The next thing that happened after that was a page fault, and then it was "all over but the crying crash dump analysis".
Even before that call took place, the few calls before that are telling:

"NTUserCreateWindowEx"

Suggests that perhaps a user had opened a program and the system was ready to draw the window.
"xxxCreateWindowEx" is the next step in the process, then it called "SetTiledRect".

Finally - Use the Blue Screen Reference in the help section!
Follow these steps to find out what a Blue Screen of Death error code means

Open WinDbg
Help –> Contents (or press F1)
Expand the "Debugging Techniques" tree.
Expand the "Bug Checks (Blue Screens)" tree.
Expand the "Bug Check Code Reference" tree.
Select the stop code referenced in the memory dump file (in my example it was 3b, so I find 0x3B)

Article Reference

Windows BSOD analysis - Uses WhoCrashed, BlueScrnView,
Wiki How - How to Read Dump Files
How to solve Windows 8 crashes in less than a minute
How to solve Windows 7 crashes in minutes
Understanding Crash Dump Files
Crash Dump Analysis Part 1: How to Install the Windows Debugger
Overview of memory dump file options
Windows feature lets you generate a memory dump file by using the keyboard
The Case of the Crashed Phone Call
How to Read Crash Reports
Windows Memory Dumps: What Exactly Are They For?
How To Exam Memory Dump File to Find the Cause of Blue Screen of Death

Tools To Help with BSOD Issues

BlueScreenView - View the Blue screen of death (STOP error) information stored in dump files
WinCrashReport - Displays a report about crashed Windows application.
WhatIsHang - Get information about Windows software that stopped responding (hang)
AppCrashView - View application crash information on Windows 7/Vista.
WhatInStartup - Enable/Disable/Delete the programs that run at Windows Startup.
WhoCrashed - WhoCrashed reveals the drivers responsible for crashing your computer
Driver Verifier - Identify issues with Windows drivers for advanced users