

QUER-TREIBER ENTLARVEN

Nur wer Windows-Fehler versteht, kann diese ausschalten. Der Beitrag erklärt die wichtigsten Fehlerklassen und zeigt die besten Lösungswege.

Oliver Ibelshäuser

Häufig werden in der Literatur die Fehlerklassen anhand der ausgegebenen Stop-Meldungen unterschieden. Das ist nachvollziehbar, aber nicht immer hilfreich: Zum einen kennt allein Windows XP mehr als drei Dutzend Stop-Kategorien. Zum anderen leistet die Beschreibung zur Stop-Message nur eine sehr vage Eingrenzung des Problems, da jeder dieser Typen wiederum für eine Vielzahl von möglichen Fehlern steht.

Effizienter dagegen ist eine Klassifizierung anhand der Ursachen, die – vereinfacht ausgedrückt – Oberkategorien der Stop-Meldungen bilden.

Interrupt-Probleme aufspüren

Bluescreens zeigen in der überwiegenden Zahl der Fälle die Meldung IRQL_NOT_LESS_OR_EQUAL. Dahinter verbirgt sich ein Interrupt-Fehler.

Hintergrund: Interrupts, die von der Hardware ausgelöst werden, sind nicht identisch mit Software-Interrupts, für die das OS oder Anwendungen verantwortlich sind. Seit Windows NT benutzt das System ein eigenes, abstraktes Modell zur Interrupt-Erfassung. Ein Dispatcher ordnet die Interrupts gemäß der von Windows vergebenen Prioritäten. Das führt zu den so genannten Interrupt Request Levels (IRQL). Je höher die Zahl, desto wichtiger ist der Interrupt für das System. User-Threads laufen mit dem IRQL 0, dem so genannten Passive Level, was der untersten Hierarchiestufe entspricht. Erhöht nun eine Anwendung den IRQL-Level, bis der für den Kernel reservierte Grad erreicht wird (Dispatch Level), ruft der Memory Manager KeBugCheckEx

```

IMAGE_NAME:  myfault.sys
DEBUG_FLR_IMAGE_TIMESTAMP:  3dd03737
STACK_COMMAND:  .trap ffffffff270bb18 ; kb
BUCKET_ID:  0xC5_myfault+347
Followup:  MachineOwner

kd> kb
ChildEBP RetAddr  Args to Child
f270bafc 804d7c50 0000000a e1c79000 00000002 nt!KeBugCheckEx+0x19
f270bafc 80536d4e 0000000a e1c79000 00000002 nt!KiTrap0E+0x2ad
f270bbe0 f8c01347 00000001 00000000 206b6444 nt!ExAllocatePoolWithTag+0x867
WARNING: Stack unwind information not available. Following frames may be wrong.
f270bc68 80571f1e 823863c0 820dd800 82390680 myfault+0x347
f270bd00 805863d5 00000008 00000000 00000000 nt!IopXXXControlFile+0x5a5
f270bd34 8044de91 00000008 00000000 00000000 nt!NtDeviceIoControlFile+0x28
f270bd34 7ffe0304 00000008 00000000 00000000 nt!KiSystemService+0xc4
0012f448 00000000 00000000 00000000 00000000 SharedUserData!SystemCallStub+0x4

```

◀ In der Mitte des Stacks wird die für den Fehler verantwortliche Systemdatei gelistet.

auf. Die Folge ist ein Bluescreen. Kernelmodus-Komponenten wie etwa Systemtreiber können nur auf andere Prozesse zugreifen, wenn diese den gleichen oder einen niedrigeren IRQL besitzen.

Mit dem Befehl `!analyze -v` in der Eingabezeile des Debuggers ermittelt der Anwender den Täter schnell: In der Regel handelt es sich um eine Treiber- oder Systemdatei. Genauere Einblicke gewähren die Kernelstack-Ausschnitte, die der Benutzer mit dem Kommando `kb` aufruft. Der Stackframe stellt eine hierarchische Liste der geladenen Module samt Adressen dar. Dort lässt sich anhand der Fehlermeldung schnell nachvollziehen, wie der Treiber schrittweise den BSOD verursacht (Bild oben).

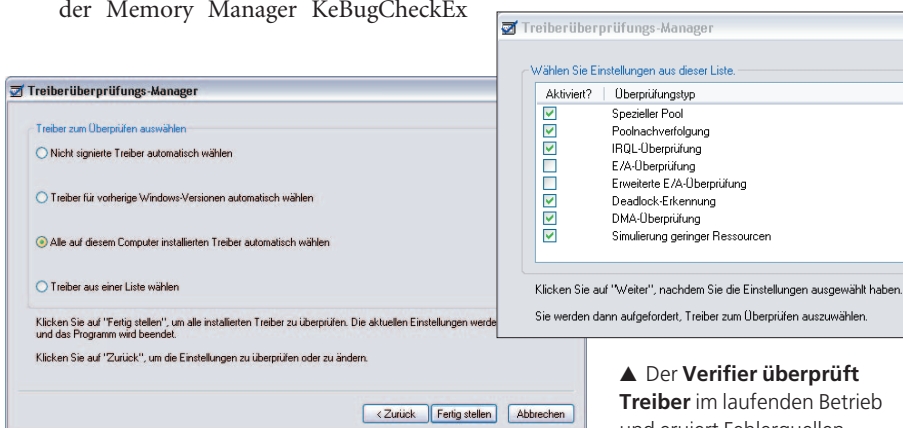
Speicherverletzungen: Buffer-Errors

Windows weist jeden Treiber an, in einem definierten Speicherbereich zu arbeiten. Unterschreitet der Treiber

diese Grenzen (*Buffer underrun*) oder überschreitet er diese (*Buffer overrun*), kann ein Fehler folgen. Genauso wahrscheinlich ist allerdings, dass das System stabil weiterläuft. Der Grund: Nur wenn der verletzte Speicherbereich von einem korrekt angeforderten Treiber oder dem Kernel selbst genutzt wird, kommt es zum Crash. Im Alltag können demnach zwischen der eigentlichen Speicherverletzung und dem Bluescreen mehrere Minuten vergehen.

Das bedeutet im gleichen Zuge, dass die bekannten Muster zur Fehleranalyse nicht mehr greifen. Im Debugger taucht als Fehlerursache vermutlich der Kernel selbst auf (`ntoskrnl.exe`). Auch der Stack-Text hilft nicht weiter, da dieser lediglich die zuletzt vom User aufgerufene Anwendung führt. Das Programm hat allerdings die Berechtigung, den entsprechenden Puffer zu nutzen, und scheidet damit als Täter aus.

Windows XP hat das passende Werkzeug allerdings bereits an Bord, den Verifier, der als `verifier.exe` im System-Ordner liegt. Der Anwender ruft das Programm mit Doppelklick auf, behält im *Treiberüberprüfungs-Manager* die Voreinstellung bei und klickt auf *Weiter*. Im Dialog *Treiber zum Überprüfen wählen*, sollte sich der Benutzer für die Option *Alle auf diesem Computer installierten Treiber automatisch wählen* entscheiden, falls keinerlei Hinweise auf den Urheber vorliegen. Mit *Fertigstellen* wird der Dialog geschlossen. Die Einstellungen sind nach dem nächsten Windows-Start aktiv. Kommt es erneut zum Bluescreen, verrät XP bereits auf dem Absturz-Bild, welches File verantwortlich war. Auch der Debug-Check



▲ Der Verifier überprüft Treiber im laufenden Betrieb und eruiert Fehlerquellen.

