Universitatea POLITEHNICA din Bucureşti

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Sisteme de Operare

 Tema de casa

 Analiza Crash Dump

 Catrina Mihai Cristian 431A

 Chiric Miruna Loredana 431A

**Cuprins**

**1.Generalitati despre Crash Dump(Catrina Mihai Cristian 431A)**

 **1.1Introducere**

 **1.2Blue Screen-ul**

**2.Depanarea Crash-urilor(Chiric Miruna Loredana 431A)**

 **2.1**Fisierele Crash Dump

 **2.2Generatia Crash Dump**

 **2.3Windows Error Reporting**

**3.Analiza Avansata a Crash-urilor(Catrina Mihai Cristian 431A)**

 **3.1Stack Trashes**

 **3.2Sisteme Hung si Unresponsive**

 **3.3Crash dump-uri false**

**4.Bibliografie**

ANALIZA CRASH DUMP-ului

**1.Generalitati despre crash dump**

**1.1Introducere**

Aproape fiecare utilizator de Windows a auzit, in cazul in care nu a experimentat de infamul "blue screen of dead ".Acest termen de rau augur se refera la ecranul albastru care este afisat atunci cand Windows-ul se blocheaza, sau se opreste din executare, din cauza unei avarii catastrofale sau o conditie interna care previne sistemul sa continue sa ruleze.

In aceasta lucrare, vom acoperi problemele de baza care provoaca Windows-ul sa se prabuseasca , vom descrie informatii despre blue screen, si se vor explica diferitele optiuni de configurare disponibile sa creeze un crash dump, o inregistrare memoriei de sistem la momentul unui crash care te poate ajuta sa iti dai seama care componenta a provocat crash-ul si de ce.

Aceast capitol va va arata, de asemenea, cum sa se analizeze un crash dump pentru a identifica un driver sau o componenta defecta. Efortul necesar pentru a efectua analize de baza crash dump este minim si ia cateva minute. Chiar daca o analiza constata problema driver-ului pentru doar unul din fiecare cinci sau zece crash dump-uri, ea totusi merita facuta: o analiza de succes poate evita pierderi viitoare de date, downtime de sistem si frustrare.

Windows-ul se blocheaza (se opreste din executie si afiseaza blue screen-ul) din mai multe motive posibile. O sursa comuna este o trimitere la o adresa de memorie care cauzeaza o incalcare de acces, fie o scriere a operatiei pentru ROM sau o operatie de citire de la o adresa care nu este mapata.[1]

O alta cauza frecventa este o exceptie neasteptata sau un trap. Crash-uri apar, de asemenea, atunci cand un subsistem kernel (cum ar fi memory manager si power manager) sau un driver (cum ar fi un USB sau un Display Driver) detecteaza incoerentele din exploatarea lor.[1]

Atunci cand un driver de dispozitiv kernel-mode sau subsistem face o exceptie ilegala, Windows-ul se confrunta cu o dilema dificila. Acesta a detectat ca o parte a sistemului de operare, cu capacitatea de a accesa orice dispozitiv hardware si orice memorie valabila a facut ceva ce nu trebuia sa faca.

Dar de ce inseamna asta ca Windows-ul trebuie sa faca crash? Nu putea sa ignore pur si simplu o exceptie si driverul de dispozitiv sau subsistem sa continue ca si cum nimic nu s-ar fi intamplat? Exista posibilitatea ca eroarea sa fie izolata si componenta sa se recupereze cumva. Dar ceea ce este mai mult probabil este ca exceptia detectata sa fi rezultat din probleme profunde, de exemplu, de la o coruptie generala a memoriei sau de la un dispozitiv hardware care nu functioneaza corect.

Permitand sistemului sa continue sa functioneze, ar rezulta probabil si mai multe exceptii, si datele stocate pe disc sau alte periferice ar putea fi defectate,acesta fiind un risc prea mare.

 De aceea Windows-ul adopta o politica ” fail fast” in incercarea de prevenire a coruptiei in RAM ,de imprastierea pe disc.

**1.2Blue Screen-ul**

 Indiferent de motivul pentru care avem un crash de sistem , functia care realizeaza efectiv crash-ul este KeBugCheckEx , documentata in Windows Driver Kit ( WDK ) . Aceasta functie are un cod de oprire ( uneori numita cod bugcheck ) si patru parametri , care sunt interpretati pe o baza de cod per -stop . Dupa mastile KeBugCheckEx dintre toate intreruperile de pe toate procesoarele din sistem , se comuta pe ecran intr-un mod grafic low - rezolutie VGA ( cel implementat de toate placile video cu suport de Windows ) , picteaza un fundal albastru , iar apoi afiseaza codul de oprire , urmat de un text, sugerand ceea ce poate face utilizatorul. In cele din urma , KeBugCheckEx cere ca orice dispozitiv driver callbacks bugcheck inregistrat ( inregistrat prin apelarea functiei KeRegisterBugCheckCallback ) , sa permita drivere-lor posibilitatea de a opri dispozitivele lor . Apoi apeleaza cauzele inregistrate callbacks ( inregistrata cu KeRegisterBugCheckReasonCallback ) , care permite driver-lor sa adauge date la crash dump sau sa scrie informatii crash dump de dispozitive alternative . ( Este posibil ca structuri de date ale sistemului sa fi fost defectate atat de serios incat ecranul albastru sa nu fie afisat . ) Figura 1 prezinta o mostra din blue screen-ul Windows-ului :

.

 FIGURA 1 Exemplu de blue screen

 KeBugCheckEx afiseaza reprezentarea textuala a codului de oprire langa partea de sus a ecranului albastru , codul numeric de oprire si patru parametri in partea de jos a ecranului albastru .[1]

 Prima linie in sectiunea cu informatii tehnice arata codul de oprire si cei patru parametri suplimentari trecuti la KeBugCheckEx . O linie de text in partea de sus a ecranului ofera un text echivalent al codului de oprire anumeric identificator. In conformitate cu exemplul din figura 1,codul de oprire 0x000000D1 este un crash DRIVER\_IRQL\_NOT\_LESS\_OR\_EQUAL . Cand un parametru contine o adresa a unei bucati din sistemul de operare sau codul device-ului driver ( ca in figura 1 ), Windows-ul afiseaza adresa de baza a modulului adresa, date stamp , precum si numele de fisier al driver-ului de dispozitiv . Aceste informatii ar putea ajuta la identificarea componentei defecte . Desi exista mai mult de 300 de coduri unice , cele mai multe sunt foarte rare , si sunt vazute pe sistemele de productie . In schimb , doar cateva coduri de oprire comune reprezinta majoritatea crash-urilor in sistem-ul Windows . De asemenea ,semnificatia celor patru parametri suplimentari depinde de codul de oprire ( nu toate codurile de stop au extinse parametrii de informatii) . Cu toate acestea , cautand codul de oprire si semnificatia parametrilor ( daca este cazul ) s-ar putea cel putin sa va ajute la diagnosticarea componentei care este lipsa ( sau a dispozitivului hardware care este cauza crash-ului ) . Puteti gasi informatii ale codului de oprire in sectiunea “Bug Checks (Blue Screens) " in fisierul ajutator Debugging Tools for Windows. Puteti cauta informatii, de asemenea si in Microsoft’s Knowledge Base (http://support.microsoft.com ) despre codul de oprire si numele de hardware suspect. S-ar putea gasi informatii despre o solutie , un update sau un pachet de servicii care rezolva orice problema pe care o ai. Fisierul Bugcodes.h contine in WDK o lista completa a celor 300 de coduri de oprire , cu unele detalii suplimentare cu privire la cauzele unora dintre ele . Pe baza datelor colectate de la lansarea Windows Vista , topul primelor 30 de coduri de oprire , justifca 96% din crash-uri si pot fi grupate intr- o gramada de categorii :

 **Page fault**- Un defect al paginii pe o memorie sustinuta de date intr- un fisier de paginare sau un fisier memorymapped are loc la un ICCV de nivelul DPC / expediere sau mai sus , care ar necesita managerul-ui de memorie sa astepte pentru ca o operatie de I / O sa apara . Kernel-ul nu poate astepta sau reprograma thread-uri la un ICCV de nivelul DPC sau mai mare Aceasta categorie include, de asemenea, defecte de pagini din zonele nepaginate . Codurilor comune de oprire sunt :

 0xA - IRQL\_NOT\_LESS\_OR\_EQUAL

0xD1 - DRIVER\_IRQL\_NOT\_LESS\_OR\_EQUAL

**Power manager**- Un driver de dispozitiv sau o functie din sistemul de operare care ruleaza in modul de kernel este intr- o stare de alimentare inconsistenta sau invalida . Cel mai frecvent , o parte componenta a reusit sa finalizeze o cerere a operatiunii de management I / O de energie in 10 minute . Aceasta categorie de crash este noua in Windows Vista . In versiunile anterioare ale sistemului de operare Windows , aceste esecuri nu erau urmate de crash-uri . Codurile de oprire sunt :

0x9F - DRIVER\_POWER\_STATE\_FAILURE

0xA0 - INTERNAL\_POWER\_ERROR

**Exceptions and traps**-Un driver de dispozitiv sau o functie din sistemul de operare care ruleaza in kernelmodul suporta o exceptie neasteptata sau un trap Codurile comune de oprire sunt:

0x1E - KMODE\_EXCEPTION\_NOT\_HANDLED

0x3B - SYSTEM\_SERVICE\_EXCEPTION 0x7E - SYSTEM\_THREAD\_EXCEPTION\_NOT\_HANDLED

 0x7F - UNEXPECTED\_KERNEL\_MODE\_TRAP

 0x8E - KERNEL\_MODE\_EXCEPTION\_NOT\_HANDLED with P1 != 0xC0000005 STATUS\_ACCESS\_VIOLATION

**Acces Violation**- Un driver de dispozitiv sau o functie din sistemul de operare care ruleaza in kernel Modul suporta o incalcare a memoriei de acces, care este cauzata fie de incercarea de a scrie la o pagina read-only sau prin incercarea de a citi o adresa care nu este mapata in prezent si, prin urmare, nu este o locatie de memorie valabila. Codurilor comune de oprire sunt:

 0x50 - PAGE\_FAULT\_IN\_NONPAGED\_AREA

 0x8E - KERNEL\_MODE\_EXCEPTION\_NOT\_HANDLED cu P1 = 0xc0000005

STATUS\_ACCESS\_VIOLATION

**Display**-Driver-ul dispozitivului de afisare detecteaza ca acesta nu mai poate controla unitatea de procesare grafica sau detecteaza o inconsistenta in managementul memoriei video . Codurile comune de oprire sunt :

0xEA - THREAD\_STUCK\_IN\_DEVICE\_DRIVER

 0x10E - VIDEO\_MEMORY\_MANAGEMENT\_INTERNAL

 0x116 - VIDEO\_TDR\_FAILURE

**Pool-**  Kernel-ul Pool manager-ului detecteaza un pool necorespunzatoar de referinţa. Codurile de oprire comune sunt:

0xC2 - BAD\_POOL\_CALLER

0xC5 - DRIVER\_CORRUPTED\_EXPOOL

**Memorey manager-** Kernel-ul managerului de memorie detecteaza o coruptie de memorie a structurilor de date de management sau o cerere nefondata de gestionare a memoriei . Codurilor comune de oprire sunt :

0x1A - MEMORY\_MANAGEMENT

 0x4E - PFN\_LIST\_CORRUPT

**Consistency check-**  Acesta este un catch-all categories pentru diverse alte verificari de consistenta efectuate de catre driver , kernel sau dispozitiv . Codurile comune de oprire sunt :

0x18 - REFERENCE\_BY\_POINTER

 0x35 - NO\_MORE\_IRP\_STACK\_LOCATION

 0x44 - MULTIPLE\_IRP\_COMPLETE\_REQUESTS

0xCE -DRIVER\_UNLOADED\_WITHOUT\_CANCELLING\_PENDING\_OPERATIONS

 0x8086 – **Acesta este un cod de oprire utilizat de catre Intel iastor.sys (driver de stocare** )

**Hardware**-  apare o eroare de hardware, cum ar fi machine check sau o intrerupere nonmaskable (NMI). Aceasta categorie include, de asemenea, eşecurile disc, atunci cand memory manager-ul incearca sa citeasca datele pentru a satisface page fault . Codurile de oprire comune sunt:

0x77 – KERNEL\_STACK\_INPAGE\_ERROR

 0x7A - KERNEL\_DATA\_INPAGE\_ERROR

 0x124 - WHEA\_UNCORRECTABLE\_ERROR

 0x101 - CLOCK\_WATCHDOG\_TIMEOUT

**USB-** O eroare irecuperabila apare intr- o operatiune Universal Serial Bus . Cod comun de oprire este :

0xFE - BUGCODE\_USB\_DRIVER

**Critical Object**- O eroare fatala apare intr-un obiect critic, fara de care Windows nu poate continua sa ruleze. Codul comun de oprire este:

0xF4 - CRITICAL\_OBJECT\_TERMINATION

**NTFS-file system-** O eroare fatala este detectata de sistemul de fisiere NTFS . Codul comun de oprire este :

 0x24 - NTFS\_FILE\_SYSTEM



 Figura 2

Figura 2 arata distribuţia acestor categorii pentru Windows Vista SP1 in septembrie 2008.

**2.Depanarea Crash-urilor**

**2.1Fisiere Crash-Dump**

În mod implicit, toate sistemele de operare Windows sunt configurate pentru a încerca înregistrarea informațiilor cu privire la starea sistemului când sistemul se blochează. Putem vedea aceste setări prin deschiderea System tool din Control Panel, făcând clic pe fila System Properties dialog box,și apoi faceți clic pe butonul Settings sub Startup și Recovery. Setările implicite pentru Sistemul de operare Windows sunt prezentate în figura 1.



**FIGURA 1**

Trei niveluri de informații pot fi înregistrate pe un accident de sistem:

* **Imagine de memorie completă** O imagine de memorie completă conține toată memoria fizică în momentul accidentului. Acest tip de dump impune ca un fișier pagină să aibe dimensiunea memoriei fizice, plus 1 MB pentru antet. Driverele de dispozitiv pot profita de până la 256 MB de datele basculante de dispozitiv, dar nu este nevoie de spațiu suplimentar pentru un antet. Deoarece aceasta poate solicita un fișier neobișnuit de mare de pagini pe sisteme mari de memorie,acest tip de fișier de imagine este setarea cel mai puțin comună. Dacă sistemul are mai mult de 2GB de RAM, această opțiune va fi dezactivată în UI, dar se poate activa manual setând CrashDumpEnabled la valoarea 1 din HKLM \ SYSTEM \ CurrentControlSet \Control \ CrashControl registry key. La momentul initializarii, Windows va verifica dacă dimensiunea paginii-fișier este suficient de mare pentru un depozit complet și trece automat la crearea unui mic depozit de memorie, dacă nu. Sistemele de servere mari s-ar putea să nu aibă un spațiu pentru un depozit complet dar poate fi în măsură să arunce informații utile, astfel încât să puteți adăuga IgnorePagefileSize la aceeași cheie de registri pentru ca sistemul sa genereaze un depozit până când rămâne fără spațiu.[2]
* **Memoria kernelului dump** O memorie a kernelului dump conține doar kernel-mode de citire /scriere pagini prezente în memoria fizică la momentul accidentului. Acest tip de depozit nu conține pagini care aparțin proceselor utilizatorului. Pentru că numai codul kernel-mode poate provoca direct Windows-ul sa se blocheze, cu toate acestea, este puțin probabil ca paginile proceselor de utilizator sa fie necesare pentru a depana un accident. În plus, toate structurile de date relevante pentru crash dump,analiza, inclusiv lista de procese care rulează, stiva de firul curent, și lista de drivere,sunt încărcate stocate în memorie nepaginate care salvează într-o memorie kernel dump

Nu există nici o modalitate de a prezice mărimea unei memorii kernel dump deoarece dimensiunea ei depinde de cantitatea de memorie kernel-mode alocată de sistemul de operare și driverele prezente pe mașină. Aceasta este setarea implicită pentru Windows Vista și Windows Server 2008.[2]

* **Imagine de memorie mică** O imagine de memorie mică, care are dimensiunea,de obicei între 128 KB și1 MB,mai este denumită un minidump sau de triage dump,care conține codul de oprire șiparametrii, lista de drivere încărcate, structurile de date care descriu procesul curent și firul (numit EPROCESS și ETHREAD), stiva kernel pentru firul care a provocat accidentul și memoria suplimentara considerate potential relevante de euristica crash dump, cum ar fi paginile referite de registrele procesorului care conțin adrese de memorie și date basculante secundare adăugate de către drivere.

Debuggerul indică faptul că are informații limitate de care dispune atunci când se încarcă un minidump,și comenzile de bază, cum ar fi procesul, care enumeră procesele active cand nu au datele de care au nevoie. Aici este un exemplu de proces executat pe un minidump:

Microsoft (R) Windows Debugger Version 6.10.0003.233 X86

Copyright (c) Microsoft Corporation. All rights reserved.

Loading Dump File [C:\Windows\Minidump\Mini100108-01.dmp]

**Mini Kernel Dump File: Only registers and stack trace are available**

...

0: kd> !process 0 0

\*\*\*\* NT ACTIVE PROCESS DUMP \*\*\*\*

**GetPointerFromAddress: unable to read from 81d3a86c**

**Error in reading nt!\_EPROCESS at 00000000**

O memori kernel dump include mai multe informații, dar trecerea la un proces diferit de adresare la mapările spațiului nu va funcționa, deoarece datele solicitate nu se află în fișierul dump. Aici este un exemplu de debugger care încarcă o memorie kernel dump, urmată de o încercare de a trece la o adresare de spatii diferita:

Microsoft (R) Windows Debugger Version 6.10.0003.233 X86

Copyright (c) Microsoft Corporation. All rights reserved.

Loading Dump File [C:\Windows\MEMORY.DMP]

**Kernel Summary Dump File: Only kernel address space is available**

...

0: kd> !process 0 0 explorer.exe

PROCESS 867250a8 ...

0: kd> .process 867250a8 ...

**Process 867250a8 has invalid page directories**

În timp ce o imagine de memorie completă este un superset de celelalte opțiuni, aceasta are dezavantajul că dimensiunea sa urmărește cantitatea de memorie fizică pe un sistem și, prin urmare, poate deveni greoi. Nu este neobișnuit pentru sistemele de astăzi de a avea mai mulți gigabytes de memorie, rezultând în fișiere crash dump care sunt prea mari pentru a fi încărcate pe un server FTP sau arse pe un CD.Deoarece codul de utilizator-mode și datele nu sunt folosite în timpul analizei de cele mai multe accidente (deoarece accidentele provin ca urmare a problemelor de memorie kernel, și structuri de date de sistem care își au locul în memoria kernel), o mare parte din datele stocate într-o imagine de memorie completă nu este relevant pentru analiza și, prin urmare, contribuie risipitor la dimensiunea unui fișier de imagine.Un dezavantaj final este că fișierul de paginare de pe volumul de încărcare (volumul cu directorul \ Windows) trebuie să fie cel puțin la fel de mare ca și cantitatea de memorie fizică de pe sistem plus pana la 365 MB.[1]

Un avantaj al minidump este dimensiunea sa mică, ceea ce îl face convenabil pentru schimbul, prin intermediul e-mailului, de exemplu. În plus, fiecare accident generează un fișier în directorul \ Windows \Minidump cu un nume de fișier unic compus din șirul "Mini", plus un număr secvență care contorizează numărul de minidumps în acea zi (de exemplu,Mini082608-01.dmp). Un dezavantaj al minidump-urilor este că, pentru a le analiza, trebuie să aveți acces la exact imaginile folosite pe sistemul care a generat dump-ul la timpul cand analizați dump-ul. (La un nivel minim, este nevoie de o copie a Ntoskrnl.exe potrivit pentru a efectua majoritatea analizelor de bază.) Acest lucru poate fi problematic dacă doriți să analizați un dump de pe un sistem diferit din sistemul care a generat dump-ul.Cu toate acestea, serverul simbol Microsoft conține imagini și simboluri () pentru toate versiunile recente de Windows, astfel încât să puteți seta calea de imagine în debugger pentru a indica serverului simbol, iar depanatorul va descărca în mod automat imaginile necesare. (Desigur, serverul simbol Microsoft nu va avea imagini pentru părți terțe ale driverelor instalate.)

Un dezavantaj mai important este că volumul limitat de date stocate în dump poate împiedica analiza eficientă.Puteți obține avantajele de minidumps chiar și atunci când configurarea unui sistem pentru a genera kernel sau crash dumps complete prin deschiderea unui dump mai mare cu WinDbg și utilizarea comandei .dump/m de a extrage un minidump. Rețineți că unMinidump este creat în mod automat, chiar dacă sistemul este setat pentru kernel dumps full.

**Notă.** Puteți utiliza comanda .dump din cadrul Livekd pentru a genera o imagine de memorie a unuisistem care poate analiza off-line fără a opri sistemul. Această abordare este utilă atunci când un sistem prezintă o problemă, dar încă furnizeaza servicii, și doriți pentru a depana problema fără a întrerupe serviciul.Imaginea dump rezultată nu este neapărat complet consecventă din cauza conținutului din diferite regiuni ale memoriei care reflectă diferite momente în timp,dar este posibil să conțină informații utile pentru o analiză.[2]

Când configurați memoriile kernel dumps, sistemul verifică dacă fișierul de paginare este suficient de mare, așa cum este descris mai devreme. Unele recomandări generale urmează în tabelul 1, dar acestea sunt numai dimensiunile estimate pentru că nu există nici o modalitate de a prezice mărimea unei memorii kernel dump. Motivul pentru care nu se poate prezice dimensiunea unei memorii kernel dump este că dimensiunea sa depinde de cantitatea de memorie kernel-mode în utilizarea de către sistemul de operare și drivere prezente pe mașină, la momentul accidentului.[3]

**TABLE 1 Default Minimum Paging File Sizes for Kernel Dumps**

|  |  |
| --- | --- |
| **System Memory Size** |  **Minimum Page File Size for Kernel Dumps** |
| < 128 MB | 50 MB |
| < 4 GB | 200 MB |
| < 8 GB | 400 MB |
| >= 8 GB | 800 MB |

Prin urmare, este posibil ca în momentul prăbușirii, fișierul de paginare sa fie prea mic pentru a organiza kernel dump.Dacă doriți să vedeți dimensiunea unui kernel dump pe sistemul dumneavoastră., forțați un accident manual, fie prin configurarea opțiunii de a vă permite să inițieze un accident manual de sistem de la consolă sau cu ajutorul funcției Notmyfault tool descrisă mai târziu în acest capitol. (Ambele abordările sunt descrise mai târziu în acest capitol.) La repornire, puteți verifica pentru a vă asigura ca kernel dump a fost generat și verificați-i mărimea pentru a evalua cât de mare este volumul fișierului de paginare.Pentru a fi conservator, pe sistemele pe 32 de biți puteți alege o dimensiune de fișier pagină de 2 GB, plus până la 356 MB, pentru ca 2 GB este maxim spațiul de adrese kernel-mode disponibil(dacă nu sunt boot cu 3GB și / sau opțiuni de boot USERVA, în cazul în care acest lucru poate fi de până la 3 GB).

Dacă nu aveți suficient spațiu pe volumul de încărcare pentru a salva fișierele memory.dmp, aveți posibilitatea să alegeți o locație de pe orice alt hard disk local, prin fereastra de dialog prezentată în figura 1.[2]

**2.2Generatia Crash Dump**

Atunci când sistemul de boot,verifică opțiunile crash dump-urilor configurate prin citirea registrului valoare HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl.Dacă un dump este configurat,face o copie a driverului miniport disc folosit pentru a scrie pentru volumul de încărcare în memorie și dă același nume ca și miniportul cu prefixul "dump\_" .De asemenea controlează componentele implicate cu scrierea unui crash dump-inclusiv copia driverului miniport disc,funcțiile I / O manager, care scriu dump-urile, și harta de unde volumul de încărcare a fișierului de paginare se află pe disc și salvează controlul. Când KeBugCheckEx executa, controlează componentele din nou și compară nou controlul cu ​​ce a obținut la pornire.Dacă nu se potrivesc, nu se scrie un crash dump, deoarece acest lucru ar putea eșua sau corupe pe discul.La o potrivire de control ,de succes, KeBugCheckEx scrie informațiile de crash dump direct la sectoarele de pe discul ocupat de fișierul de paginare, ocolind sistemul de fișiere a driverului și depozitele de stive ale driverului (care ar putea fi corupt sau chiar au provocat accidentul).[1]

**Notă**. Deoarece fișierul pagina este creat la începutul pornirii sistemului în timpul managerului de memorie de initializare,cele mai multe accidente sunt cauzate de bug-uri la un driver de sistem de pornire rezultat initial într-un fișier dump.Avarii la componentele timpurii de boot Windows, cum ar fi HAL sau inițializarea de drivere de boot apar prea devreme pentru ca sistemul să aibă un fișier pagină, astfel încât folosind un alt calculator pentru a depana procesul de pornire este singura modalitate de a efectua analize ale dump-urilor, în aceste cazuri.

Când Session Manager (SMSS) re-initializeaza fișierul pagină în timpul procesului de boot, se solicită funcția SmpCheckForCrashDump, care arată în fișierul de paginare curent volumul de încărcare(creat de kernel în timpul procesului de boot), pentru a vedea dacă un crash dump este prezent.SMSS apoi verifică dacă fișierul țintă dump este pe un volum diferit decât fișierul de paginare.Dacă este așa,se redenumește fișierul de paginare pentru cu un nume temporar, Dumpxxx.tmp (unde xxx este valoarea actuală scăzută a numărului de bug-uri ale sistemului), și trunchiază fișierul la dimensiunea de la dump.(Această informație este stocată în antet în partea de sus a fiecărui fișier de imagine.)De asemenea, elimină atât atributele ascunse și de sistem de fișiere. SMSS creează apoi registre cheie volatile HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl \ MachineCrash și magazine cu numele temporar al fișierului de imagine în valoare "dumpfile".Acesta scrie apoi un REG\_DWORD la"TempDestination" valoare care să indice dacă locația fișierului dump este doar destinație temporară.Dacă fișierul de paginare se află pe același volum ca și fișierul de destinație al dump-ului, temporar fișierul de dump nu este utilizat, și fișierul de paginare este redenumit direct cu numele fișierului dump. În acest caz, valoarea fișierului dump va fi %SystemRoot% \ Memory.dmp și TempDestination va fi 0.[1]

Mai târziu, în boot, verificările Wininit pentru prezența cheii MachineCrash, și dacă există,Wininit lansează WerFault, care prevede TempDestination și valorile fișierului dump și fie redenumește sau copiază fișierul temporar în locația țintă (de obicei% rădăcină de sistem% \Mem ory.dmp, dacă nu este configurat altfel), dacă ținta este în același volum ca și directorul Windows. WerFault scrie apoi în final numele de fișier de imagine la valoarea locație finale a fișierului dump în cheia MachineCrash.Aceste etape sunt prezentate în figura 2.[1]

****

**FIGURE 2** Crash dump file generation

Pentru a sprijini mașinile care nu ar putea avea un fișier de paginare sau un fișier de paginare pe volumul de încărcare,de exemplu, pe sistemele boot de pe un SAN sau read-only mass-media, Windows, de asemenea,sprijină utilizarea unui fișier dump dedicat care este configurat în DedicatedDumpFile și valorile DumpFileSize sub HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl registry-key.Când este specificat un fișier dump dedicat, driverul crash dump (% SystemRoot% \System32 \ drivers \ Crashdmp.sys) creează fișierul dump de dimensiunea dorită și scrie datele privind accidentele acolo în loc de fișierul de paginare.În cazul în care un kernel dump complet este configurat, dar nu există suficient spațiu pe volumul țintă pentru a crea fișierul dump dedicat, sistemul revine la scris un minidump.

**2.3 Windows Error Reporting**

După cum sa menționat în capitolul 3, Windows include o facilitate numit Windows Error Reporting(WER), care facilitează transmiterea automată a proceselor și eșecurile sistemului (cum ar fi accidente și / sau se blochează) la Microsoft (sau un server intern de raportare a erorilor) pentru analiză.Acestă caracteristică este activată în mod implicit, dar poate fi modificată prin schimbarea comportamentului WER-ului, care ia pas suplimentar de a determina dacă sistemul este configurat pentru a trimite un accident de dump la Microsoft (sau un server privat, explicat în detaliu în "Crash Analysis online"mai târziu în acest capitol), pentru a analiza o repornire în urma unui accident.Ecranul WER Advanced Settings, care poate fi acces de la Problem Reports și ecranul Solutions din Control Panel’s System applet,sunt arătate în figura 3.Această casetă de dialog vă permite să configurați setările de raportare a erorilor de sistem.[2]



**FIGURE 3** Error reporting configuration dialog box

Așa cum am menționat mai devreme, în Wininit.exe găsești HKLM \ SYSTEM \ CurrentControlSet \ Control \CrashControl \ MachineCrash key, se execută WerFault.exe cu-K-c steaguri (drapel k indică rapoartele de erori ale kernelului , iar steagul C indică faptul că kernel dump complet ar trebui să fie convertit la un minidump) de a avea control WerFault.exe pentru un fișier kernel crash dump.WerFault ia următoarele măsuri pentru pregătirea de a trimite un raport de accident de dump la Microsoft OnlineCrash Analysis (OCA), site-ul (sau, dacă este configurat, un server intern de raportare a erorilor):

1. Dacă tipul de dump nu a fost generat un minidump, acesta extrage un minidump din dosar și îl stochează în locația implicită de \ Windows \ Minidumps, dacă e configurat altfel prin valoarea MinidumpDir în HKLM \ SYSTEM \CurrentControlSet \ Control \ CrashControl \ key.

2. Se scrie numele fișierelor Minidump la HKLM \ SOFTWARE \ Microsoft \ Windows \Windows Error Reporting \ KernelFaults \ Queue.

3. Aceasta adaugă o comandă pentru a executa WerFault.exe (\ Windows \ System32 \ WerFault.exe) aHKLM \ SOFTWARE \ Microsoft \ Windows \ CurrentVersion \ RunOnce, astfel încât WerFault este executat o dată în timpul log-ării primului utilizator la sistem în scopul trimiterii raportului de eroare.

Când utilitarul WerFault execută timpul de conectare, ca urmare de a fi configurate în sine pentru a începe, se verifică HKLM \ SOFTWARE \ Microsoft \ Windows \ Windows Error Reporting \KernelFaults \ Queue key pentru a căuta rapoarte din coada de așteptare, care s-au adăugat în ultimele dump-uri faza de conversie. Se verifică, de asemenea, dacă există rapoarte de avarie netrimise anterior din sesiunile anterioare.Dacă există, se lansează WerFault.exe cu-K-q steaguri (drapel q specifică utilizarea modului de raportare în așteptare) pentru a genera un fișier în format XML care conține o descriere de bază a sistemului, inclusiv versiunea sistemului de operare, o listă de drivere instalate pe aparat, și lista de drivere Plug and Play încărcate pe sistemul de la momentul accidentului.

Dacă configuram pentru a cere datele introduse de utilizator (care nu este implicit), se prezintă apoi caseta de dialog(se arată în figura 4), care întreabă utilizatorul dacă el sau ea vrea să trimită un raport de eroare la Microsoft. Dacă utilizatorul alege să trimită raportul de eroare, și dacă nu prejudiciază GroupPolicy, WerFault trimite fișierul XML și minidump la http://oca.microsoft.com, care transmite datele de la o fermă de servere de analiză automată, descrise în secțiunea următoare.



**FIGURE 4** Crash dump error reporting dialog box

Ferma de servere care analizează automat folosește același motor de analiză pe care depanatoarele kernel Microsoft le folosește atunci când încărcați un fișier crash dump(descrise pe scurt).Analiza generează un ID găleată, care este o semnătură care identifică un anumit tip de accident.Ferma de servere intreaba o bază de date utilizând ID-ul găleată pentru a vedea dacă o rezoluție a fost găsită de accident, și-l trimite la o adresă URL înapoi la WerFault care se referă la OCA Website (<http://oca.microsoft.com>).Dacă este configurat să facă acest lucru, WerFault lansează Windows ErrorConsola de raportare, sau WerCon (% SystemRoot% \ System32 \ Wercon.exe), care este un program care permite utilizatorilor să interacționeze cu WER pentru a primi rezolvarea problemelor și informațiile de urmărire, precum și pentru configurarea comportamentului WER.Atunci când navigarea pentru soluții, WerCon conține un cadru browser de Internet pentru a deschide pagina de pe site-ul Web WER care raportează analiza accidentului preliminară.Dacă este disponibilă o rezolvare, pagina instruiește utilizatorul pentru a obține o remediere rapidă, service pack sau update driver terț.[3]

**3.Analiza avansata a crash-urilor**

**3.1Stack Trashes**

Stack overrun sau stack trashing rezulta de obicei din rezultatele de la un buffer overrun sau underrun sau cand un driver trece de la buffer address situata pe stiva unui driver inferior de pe device stack, care efectueaza treaba asincron.[1]

In cazul unui buffer overrun si underrun , in loc sa stea in pool, asa cum s-a intamplat cu Notmyfault’s buffer overrun bug ,tinta este stiva firului care executa bug-ul. Acest tip de bug este diferit si este dificil de depanat, deoarece stiva este fundatia pentru orice analiza crash dump..[2]

In cazul trecerii buffer-urilor de pe stiva pe drivere inferioare ,daca driverele inferioare se intorc imediat apelantului, deoarece a utilizat o rutina de finalizare pentru a efectua acest lucru, in loc sa returneze sincron , cand rutina de finalizare este apelata, se va utiliza adresa stiva care a fost trecuta in prealabil si care ar putea corespunde acum la o stare diferita de a apelantului stiva si duce la coruptie.[1]

Cand executati Notmyfault si selectati Stack Trash, overrun-ul driver-elor Myfault se aloca pe kernel stack-ul firului care le executa. Cand Myfault incearca sa returneze controlul la functia Ntoskrnl care a fost invocata, citeste adresa retur, care este adresa la care aceasta ar trebui sa continue executarea din stiva. Adresa a fost corupta de stackbuffer overrun, astfel incat firul continua executarea la alta adresa in memorie-o adresa, care ar putea sa nu contina nici macar cod. O exceptie ilegala si un crash apar atunci cand un fir executa o instructiune ilegala a procesorului sau la care face referire o memorie invalida.

Driver-ul care analizeaza crash dump-ul unui stack overrun indica vina care va varia de la un crash la altul si codul de oprire va fi aproape intotdeauna KMODE\_EXCEPTION\_NOT\_HANDLED.

Daca executati o analiza detaliata, trasarea stivei arata astfel:

STACK\_TEXT:

KeBugCheckEx 0 x1e 881fc744 81c82590 0000008e C0000005 00000000 nt!

881fcb14 81ca45da 881fcb30 00000000 881fcb84 NT! KiDispatchException 0 x1a9

881fcb7c 81ca458e 881fcc44 00000000 badb0d00 NT!CommonDispatchException 0 x4a

881fcc2c 81d07fd3 9762b658 84736e68 84736e68 NT! Kei386EoiHelper +0 x186

881fcc44 81e98615 99321810 84736e68 84736ed8 nt!IofCallDriver+0x63

881fcc64 81e98dba 9762b658 99321810 00000000 nt!IopSynchronousServiceTail+0x1d9

881fcd00 81e82a8d 9762b658 84736e68 00000000 nt!IopXxxControlFile+0x6b7

881fcd34 81ca3a1a 0000007c 00000000 00000000 nt!NtDeviceIoControlFile+0x2a

881fcd34 779e9a94 0000007c 00000000 00000000 nt!KiFastCallEntry+0x12a

WARNING: Frame IP not in any known module. Following frames may be wrong.

0012f9f4 00000000 00000000 00000000 00000000 0x779e9a94

Observati cum apelul la IofCallDriver duce imediat la Kei386EoiHelper si la o exceptie,in loc de driverul IRP-ului cu expediere de rutina. Aceasta este in concordanta cu stiva fiind corupt si IRP dispatch routine provoaca o exceptie atunci cand se incearca o intoarcere la apelant prin referire la o adresa de retur corupta. Din pacate, mecanismele ,in special pool-ul si codul sistemului de protectie la scriere nu pot rezolva acest tip de bug. Solutia este sa faceti o analiza a pasilor manual pentru a determina in mod indirect care driver a operat la momentul coruptiei. O modalitate este de a examina IRPs-urile care sunt in curs de desfasurare pentru firul care a fost executat la momentul stak trash-ului. Cand un fir emite o cerere I / O, managerul de I / O stocheaza un pointer la IRP-ul respectiv pe lista IRP a structurii ETHREAD pentru fir.Thread debugger comanda caderile in lista IRP a firului tinta.Puteti sa vizualitzati IRP-ul cu comanda *!irp:*

lkd> !thread

THREAD 858d1aa0 Cid 0248.02c0 Teb: 7ffd9000 Win32Thread: ffad4e90 RUNNING on processor 0

IRP List:

**bc5a7f68**: (0006,0094) Flags: 00000000 Mdl: 00000000

Not impersonating

Attached Process 84f45d90

...

lkd> !irp **bc5a7f68**

Irp is active with 1 stacks 1 is current (= 0x837a7ab8)

No Mdl Thread 858d1aa0: Irp stack trace.

cmd flg cl Device File Completion-Context

>[ e, 0] 0 0 856f6378 8504f290 00000000-00000000

**\Driver\MYFAULT** Args: 00000000 00000000 83360010 00000000

 Iesire arata ca locatia actuala a stack IRP-ului (marcate cu ">" prefix) este detinuta de catre driver-ul Myfault. Daca acesta ar fi un crash real, urmatorii pasi vor fi sa te asiguri ca versiunea de driver instalata este cea mai recent disponibila,apoi instalati noua versiune, daca aceasta nu este, si daca este, pentru a permite Driver Verifier pe driver (cu toate setarile, cu exceptia low memory simulation).[1]

Analiza manuala a stivei este de multe ori cea mai puternica tehnica atunci cand se ocupa cu crash-uri cum ar fi acestea. De obicei, aceasta implica dumping-ul actualului registru stack pointer (de exemplu, ESP si RSP pe 32 de biti si x 64, respectiv). Cu toate acestea, deoarece codul responsabil pentru crashing-ul sistemului in sine ar putea modifica stiva intr-un mod care face analiza dificila, procesorul responsabil pentru crash-ul sistemului ofera un magazin de suport pentru datele curente din stiva, numit KiPreBugcheckStackSaveArea, care contine o copie a stivei inaintea oricarui cod executat in KeBugCheckEx . Prin utilizarea comenzii DPS (dump pointer cu simboluri) in debugger, puteti arunca acest domeniu (in locul resgistrului stack pointer al CPU-ului) si rezolva simbolurile intr-o incercare de a descoperi orice stack trace-uri posibile. In acest crash , se arata ceea ce cade in de zona de stiva pe un sistem pe 32 de biti.[2]

KD> DPS KiPreBugcheckStackSaveArea KiPreBugcheckStackSaveArea 3000

81d7dd20 881fcc44

81d7dd24 98fcf406 myfault +0 x406

81d7dd28 badb0d00

Desi aceste date se afla printre multe alte functii diferite, este de interes special

deoarece mentioneaza o functie in driverul Myfault, care, dupa cum am vazut a executat un Irp, care nu este aratat pe stiva. Pentru mai multe informatii privind analiza manuala a stivei uita-te in Debugging Tools for Windows help file si resursele suplimentare la care voi face referire mai tarziu in acest capitol.[2]

**3.2Sisteme hung si unresponsive**

Daca un sistem nu mai raspunde (nu mai primiti nici un raspuns de tastatura sau de intrare a mouse-ului), mouse-ul se blocheaza, sau puteti muta mouse-ul, dar sistemul nu raspunde la clicuri, se poate spune ca sistemul este hung. O serie de lucruri poate provoca sistemul sa fie hang cum ar fi:

 Un driver de dispozitiv nu se intoarce de la serviciu de intrerupere (ISR) sau un apel de procedura(DPC)de rutina amanat[1]

Un impas (atunci cand doua threaduri sau procesoare de operator, care detin resurse reciproc si nici unul nu va da ceea ce are) care apare in modul kernel

Puteti verifica aparita blocajelor folosind optiunea Driver Verifier numit si detector de impasuri.Detectarea de impas monitorizeaza utilizarea de spinlock-uri, mutecsi rapizi , si mutecsi, in cautarea de modele care ar putea duce la un impas. Daca este gasit, Driver Verifier blocheaza sistemul cu o indicatie a driver-ului care provoaca impasul. Cea mai simpla forma de impas apare atunci cand doua thread-uri detin resurse reciproc fiecare thread nu vrea sa dea sau sa astepte ceea ce vrea unul dintre ele. Primul pas de depanare al sistemelor hung este de a active detectorul de impas pe driverele suspecte , drivere nesemnate, si apoi toate driverele, pana candse va obtine un crash care indica driver-ul care provoaca impasul..[2]

Exista doua moduri de a aborda un sistem hang , astfel incat sa puteti aplica tehnici manuale de depanarea crash-urilor descrise in acest capitol pentru a determina ceea ce driver-ul sau componenta cauzeazaa Hang-ul: prima este de a face un crash sistemului hung si sa speram ca vom obtine un crash dump pe care il putem analiza, iar al doilea este de a sparge sistemul cu kernel debugger si de a analiza activitatea sistemului, Ambele abordari necesita configurare inainte si un reboot. Putem utiliza aceeaasi explorare a starii sistemului cu ambele abordari pentru a determina cauza hang-ului.[1]

Pentru a bloca manual un sistem hang , trebuie mai intai sa adaugati in registru DWORD valoarea HKLM \SYSTEM \ CurrentControlSet \ Services \ i8042prt \ Parameters \ CrashOnCtrlScroll si setati-o la 1.

Dupa rebootare driver-ul i8042 portului, care este driverul portului de intrare al tastaturii PS2, monitorizeaza ,intrarile de la tastatura in ISR ,cautand doua apasari ale scroll key-ului in timp ce tasta control dreapta este apasata. Cand driverul vede secventa, aceasta apeleaza KeBugCheckEx cu MANUALLY\_INITIATED\_CRASH (0xE2) codul de oprire care indica initierea unui crash manual. Cand sistemul rebooteaza, deschide fisierul de crash dump si se aplica tehnicile mentionate mai devreme pentru a determina de ce sistemul a este hung (de exemplu, pentru a determina thread-ul executat atunci cand sistemul este hung , ceea ce indica kernel stack ca s-a intamplat, si asa mai departe). Retineti ca acest lucru functioneaza in cele mai multe scenarii de sistem Hung, dar nu va functiona daca ISR driverul de port i8042 nu se executa. (ISR i8042 driverul de port nu se va executa daca toate procesoarele sunt hang ca rezultat IRQL-ul acestora este mai mare decat IRQL-ul ISR- lui, sau daca coruptia de structuri de date din sistem se extinde la codul de intrerupere sau la date.)[2]

Se poate declansa, de asemenea, un crash in cazul in care hardware-ul are un built-in pe butonul "crasht". (Unele high-end servere au asta.) In acest caz, accidentul este initiat de semnalizare de intrerupere Nonmaskable (INM), PIN-ul placii de baza a sistemului. Pentru a activa aceasta, setati valoarea DWORD de registry HKLM \ SYSTEM \ CurrentControlSet \ Control \ CrashControl \ NMICrashDump la 1. Apoi, cand apasati dump switch NMI este livrat cu sistemul si kernel-ul NMI intrerupe apelurile handler-ului KeBugCheckEx. Aceasta functioneaza in mai multe cazuri decat mecanismul driverului de port i8042 deoarece NMI IRQL este intotdeauna mai mare decat cel al port-ului driver-ului de intrerupere.

Daca nu sunt in masura sa genereze manual un crash dump , puteti incerca sa spargeti sistemul hung prin setarea ,in primul rand a sistemului de boot in modul de depanare. Puteti face acest lucru intr-unul din doua moduri.: aveti posibilitatea sa apasati tasta F8 in timpul de boot si sa selectati modul de depanare, sau puteti creea o optiune de boot de depanare-modului in BCD prin copierea unei intrarii de boot existente si adaugarea optiunii de debug . Atunci cand se utilizeaza abordarea F8, sistemul va utiliza implicit conexiune (Serial Port COM2 si 19200 Baud), dar puteti folosi tasta F10 pentru a afisa Edit Boot Options pentru a edita optiunile de boot legate de debug . Optiunea de debug , va necesita , configurarea mecanismului de conexiune in timp ce sistemul gazda ruleaza kernel debuggerul si sistemul tinta pornit in modul de debug si apoi configurati de bugport si de switch-urile baudrate corespunzatoare pentru tipul de conexiune.Cele trei tipuri de conexiuni sunt un modem cu cablu nul , folosind un port serial, un IEEE 1394 (FireWire) cablu ce foloseste 1394 porturi de pe fiecare sistem, sau un cablu host-to-host USB 2.0 folosind porturile USB pe fiecare sistem. [1]

La pornirea in modul de depanare, sistemul incarca depanatorul kernel in momentul de boot si il pregateste pentru o conexiune de la un kernel debugger care ruleaza pe un alt computer conectat printr-un cablu serial, cablu IEEE 1394, sau cablu host-to-host USB 2.0. Retineti ca prezenta kernel debugger-ului nu afecteaza performantele. Cand sistemul este hung porniti WinDbg sau Kd debugger-ul asupra sistemului conectat, sa stabileasca o conexiune kernel debugger , si sa sparga sistemul hung. Aceasta abordare nu va functiona daca intreruperile sunt dezactivate sau depanatorul kernel-ului a devenit corupt.

In loc sa parasiti sistemul in halted state in timp ce efectuati analiza, puteti utiliza, comanda .dump a debugger-ului pentru a crea un fisier crash dump pe masina host debugger.Apoi, aveti posibilitatea sa reporniti sistemul hang si sa analizaticrash dump-ul offline . Retineti ca acest lucru poate dura o lunga perioada de timp, daca sunteti conectat cu ajutorul unui modem nul de serie cablu sau o conexiune USB 2.0 (fata de o conexiune de mare viteza de 1394), asa ca poate doriti sa capturati doar un minidump folosind. comanda dump / m.. Alternativ, in cazul in care masina tinta este capabila sa scrie un crash dump, o puteti obliga s-o faca prin comanda .crash din debugger. Acest lucru va duce la creearea unui dump de masina tinta pe hard drive-ul local al unitatii pe care o puteti examina dupa repornirea sistemului. [1]

Veti putea provoca un hang , pornind Notmyfault si selectand optiunea Hang. Aceasta provoaca Myfault driver catre coada de DPC pe fiecare procesor de sistem care executa o bucla infinita.Deoarece IRQL-ul procesorului executa functii de DPC este nivelul DPC / expediere, tastatura ISR va raspunde la secventa speciala de crashing a tastaturii.[2]

Dupa ce ati intrat intr-un sistem hang de unul singur sau ati incarcat un dump generat manual de la un sistem hung intr-un program de depanare, trebuie sa executati comanda !analyze cu optiunea -Hang.Acest lucru face ca debugger-ul sa examineze incuietorile de sistem si sa incerce sa determine daca exista un impas, si daca da, ce driver sau drivere sunt implicate. Cu toate acestea, pentru o un hang ca acesta generat cu optiunea Notmyfault’s Hang , comanda de analiza !analyze nu va raporta nimic util. [2]

In cazul in care comanda !analyze nu identifica problema, executa !thread si !process vezi in fiecare context de dump al CPU-ului ceea ce face fiecare procesor. In cazul in care un thread are sistemul hung de executare intr-o bucla infinita la un IRQL de nivelul DPC / expediere sau mai mare, veti vedea vedea modulul driver-ului care s-a blocat in stack trace –ul comenzii !thread. Stack trace-ul crash dump-ului care se obtine atunci cand crash de sistem se confrunta cu Hang bug Notmyfault si arata ca acest lucru:

STACK\_TEXT:

f9e66ed8 f9b0d681 000000e2 00000000 00000000 nt!KeBugCheckEx+0x19

f9e66ef4 f9b0cefb 0069b0d8 010000c6 00000000 i8042prt!I8xProcessCrashDump+0x235

f9e66f3c 804ebb04 81797d98 8169b020 00010009 i8042prt!I8042KeyboardInterruptService+0x21c

f9e66f3c fa12e34a 81797d98 8169b020 00010009 nt!KiInterruptDispatch+0x3d

WARNING: Stack unwind information not available. Following frames may be wrong.

ffdff980 8169b288 f9e67000 0000210f 00000004 myfault+0x34a

8054ace4 ffdff980 804ebf58 00000000 0000319c 0x8169b288

8054ace4 ffdff980 804ebf58 00000000 0000319c 0xffdff980

8169ae9c 8054ace4 f9b12b0f 8169ac88 00000000 0xffdff980

...

Primele cateva randuri de referinta ale stack trace-ului se executa atunci cand tastati secventa crash key a port driver-ului i8042. Prezenta driver-ului Myfault indica faptul ca acesta ar putea fi responsabil pentru hang. O alta comanda care ar putea fi utila este !lacks . In mod implicit, comanda listeaza numai resursele care sunt intr-o disputa, ceea ce inseamna ca ambele sunt detinute si asteapta cel putin un thread pentru a le achizitiona. Examinati thread-urile stivei detinute de proprietari cu comanda ! *thread*  pentru a vedea ce driver se executa .Uneori, veti gasi ca proprietarul unuia dintre lacate asteapta ca IRP sa se completeze (o lista de IRP-uri in legatura cu un thread-ul afisat la iesirea !thread). In aceste cazuri este foarte greu de spus de ce un IRP-ul nu progreseaza. Un lucru pe care il poti face sa examinezi IRP este comanda !irp si sa gasiti drverul care asteapta decizia IRP-ului.Odata ce aveti numele driverului, puteti folosi comanda !stacks pentru verifca alte thread-uri pe care driver-ul le-ar putea rula, care ofera de multe ori indicii despre ceea ce driverul lacatului detinut face. O mare parte din timp, veti gasi driverul in impas sau astepta nd o alta resursa care este blocata asteptand driver-ul.[1]

**3.3Crash dump-uri false**

In aceasta parte , vom aborda modul de depanare a sistemelor care din anumite motive nu sunt inregistrate ca un crash dump. Un motiv pentru care un crash dump ar putea sa nu fie inregistrat este in cazul in care paginarea fisierului pe volumul de incarcare este prea mica pentru a detine crash dump-ul . Acest lucru poate fi remediat cu usurinta prin cresterea dimensiunii fisierului de paginare. Un al doilea motiv pentru care nu ar putea fi un crash dump inregistrat se datoreaza codului kernel si structurilor de date ce necesita scrierea in crash dump –ul care a fost corrupt in timpul crash-ului. Asa cum este descris mai devreme, aceste date sunt rezumate sumar atunci cand sistemul booteaza, iar daca sumarizarea nu se potriveste momentul crash-ului , sistemul nici macar nu incearca sa salveze crash dump-ul (pentru a nu risca deteriorarea datelor pe disc). Deci, in acest caz, aveti nevoie pentru a prinde momentul in care sistemul se prabuseste si apoi sa incercati sa determinati motivul crash-ului.[3]

Un alt motiv este produs atunci cand subsistemul discului pentru discul de sistem nu este capabil sa proceseze cereri de scriere pe disc (o conditie care ar putea fi declansat esecul sistemului in sine). O astfel de conditie ar fi o problema hardware in controlerul discului sau, poate, o problema de cablare in apropierea hard disk-ului.[1]

Cu toate acestea, o alta posibilitate care apare atunci cand sistemul are driverele care s-au inregistrat pentru a adauga datele secundare dump la fisierele dump . Cand callback-ul driverelor sunt apelate , ar putea accesa incorrect structuri de date situate in pagina de memorie), ceea ce va conduce la un al doilea crash.[1]

O optiune simpla este de a dezactiva optiunea Automatically Restart in optiunile Start Up and Recovery si astfel cand sistemul face crash , se poate examina blue screen-ul de pe consola.Cu toate acestea, numai crash-urile cele mai simple pot fi rezolvate doar de textul blue screen-ului .[2]

Pentru a efectua mai multe analize avansate , trebuie sa utilizati kernel debugger-ul si sa vizualizati sistemul in momentul crash-ului . Acest lucru poate fi realizat prin bootarea sistemului in modul de debug, care a fost descris anterior. Atunci cand un sistem este pornit in modul de depanare si face crash , in loc de blues screen si incercarea de a inregistra crash dump-ul , va astepta la nesfarsit pana cand un kernel debugger-ul va fi conectat. In acest fel, puteti vedea motivul crash-ului si puteit face efectuarea unei analize de baza utilizand comenzile kernel debugger-ului descris mai devreme. Dupa cum s-a mentionat anterior, aveti posibilitatea sa utilizati comanda .dump in debugger pentru a salva o copie a spatiului memoriei de sistem pentru a face debugging-ul mai tarziu, astfel permitandu-va sa reporniti sistemul crash si sa depanati offline problema.[1]

Codul sistemului de operare si structurile de date care se ocupa de exceptiile procesorului pot deveni deteriorate astfel o serie de defecte recursive. Un exemplu in acest sens ar fi daca apar sisteme de operare handler capcana, care au devenit corupte si au cauzat un page fault . Acesta va indica page fault handler-ul care ar fi de vina din nou si asa mai departe . Daca o astfel de situatie a avut loc, sistemul ar fi fara speranta blocat. Pentru a preveni o astfel de situatie care apare la CPU, procesoarele au un mechanism de built recursiv ,care stabileste o limita greu de atins un fault recursiv . Pe cele mai multe procesoare x86, un defect poate gazdui pana la doua niveluri de adancime. Cand un fault apare , se reseteaza procesorul in sine si se reporneste masina. Aceasta se numeste un defect triplu. Acest lucru se poate intampla atunci cand exista o componenta hardware defect.Nici macar un kernel debugger nu va fi invocat intr-o situatie de defect triplu. Cu toate acestea, uneori, simplul fapt ca kernel-ul debuggerului nu se activeaza poate confirma ca exista o problema cu o noua adaugare a hardware-ului sau driverelor[1]

**4.Bibliografie**

1.Windows Internals Fifth Edition-Mark E. Russinovich

and David A. Solomon with Alex Ionescu

2.Windows Internals Sixth Edition - Mark E. Russinovich

and David A. Solomon with Alex Ionescu

3. Optimizing Crash Dump in Virtualized Environments-Yijian Huang, Haibo Chen, Binyu Zang