

Add `_except_handler4_common` to MSVCRT.DLL

msvcrt-except_handler4_common

Purpose/History

Software intended to run on later than Windows 8/2012, does use `_except_handler4_common` function in `MSVCRT.DLL`. These steps allow you to add it.

This function is used for error handling in application in Windows 8/2012 and later.

These files come from the Microsoft Visual C++ Runtime Source Code that comes with Visual Studio 2017. The actual error handler code comes from https://github.com/Speedi13/ManualMapped_SEH_32bit

Requirements

1. A x86 version of Windows 2003 / XP SP1 with source code available: see [build-win2k3](#)
2. Recommended: version check unlock to allow run software for Windows Vista and above: [kernel32-version-unlock](#)
3. `nullptr` emulation - new feature in C++ not supported in these build tools: [extras-nullptr](#)
4. `\base\crts\crtw32\extras` folder - where .c/.h files are stored that will be added to MSVCRT.DLL when compiled and linked

Changes

1. Add `gs_cookie.c` `gs_report.c` `gs_support.c` from the Microsoft Visual C++ Runtime Source Code that comes with Visual Studio 2017 or from below links the `\base\crts\crtw32\extras` folder:
https://public-archive.org/hacked_team/rcs-dev%5Cshare/HOME/cod/CRT/crt/src/vcruntime/gs_cookie.c
https://public-archive.org/hacked_team/rcs-dev%5Cshare/HOME/cod/CRT/crt/src/vcruntime/gs_report.c
https://public-archive.org/hacked_team/rcs-dev%5Cshare/HOME/cod/CRT/crt/src/vcruntime/gs_support.c
2. Add the following `chandler4.cpp` to the `\base\crts\crtw32\extras` folder:

```

#include <Windows.h>
#ifdef _X86_
#error Only on 32bit it works this way
#endif

#ifdef _MSC_VER < 1600 //MSVC version <8
#include <nullptr_emulation.h>
#endif

typedef EXCEPTION_DISPOSITION NTAPI EXCEPTION_ROUTINE (struct _EXCEPTION_RECORD *ExceptionRecord, PVOID
EstablisherFrame, struct _CONTEXT *ContextRecord, PVOID DispatcherContext);
typedef EXCEPTION_ROUTINE *PEXCEPTION_ROUTINE;

typedef struct _EXCEPTION_REGISTRATION_RECORD
{
    struct _EXCEPTION_REGISTRATION_RECORD* Next;
    PEXCEPTION_ROUTINE Handler
} EXCEPTION_REGISTRATION_RECORD, *PEXCEPTION_REGISTRATION_RECORD; /* size: 0x0010 */

//to access the global PE variables:
extern "C" LPVOID __ImageBase;
extern "C" ULONG_PTR __security_cookie;

void* g_ImageStartAddr = nullptr;
void* g_ImageEndAddr = nullptr;

//the exception handler function above works for other exceptions but is labeled only for SEH4 since its easier to
understand that way
//so this is a more generically named function
LONG NTAPI ExceptionHandler(_EXCEPTION_POINTERS *ExceptionInfo)
{
    //making sure to only process exceptions from the manual mapped code:
    PVOID ExceptionAddress = ExceptionInfo->ExceptionRecord->ExceptionAddress;
    if ( ExceptionAddress < g_ImageStartAddr || ExceptionAddress > g_ImageEndAddr )
        return EXCEPTION_CONTINUE_SEARCH;

    DWORD RegisterESP = ExceptionInfo->ContextRecord->Esp;
    EXCEPTION_REGISTRATION_RECORD* pFs = (EXCEPTION_REGISTRATION_RECORD*) __readfsdword( 0 ); // mov pFs, large fs:0 ;
    <= reading the segment register
    if ( (DWORD_PTR)pFs > (RegisterESP-0x10000) && (DWORD_PTR)pFs < (RegisterESP+0x10000) ) //validate pointer
    {
        //pp(41) : error C2440: 'initializing' : cannot convert from 'void *' to 'EXCEPTION_ROUTINE (__stdcall *)'
        EXCEPTION_ROUTINE* ExceptionHandlerRoutine = pFs->Handler;
        if ( ExceptionHandlerRoutine > g_ImageStartAddr && ExceptionHandlerRoutine < g_ImageEndAddr ) //validate
pointer
        {
            EXCEPTION_DISPOSITION ExceptionDisposition = ExceptionHandlerRoutine( ExceptionInfo->ExceptionRecord, pFs,
ExceptionInfo->ContextRecord, nullptr );
            if ( ExceptionDisposition == ExceptionContinueExecution )
                return EXCEPTION_CONTINUE_EXECUTION;
        }
    }
    return EXCEPTION_CONTINUE_SEARCH;
}

struct EXCEPTION_REGISTRATION
{
    unsigned int prev;
    unsigned int handler;
};
#pragma endregion EHStructs

typedef void (__fastcall *PCOOKIE_CHECK)(UINT_PTR);

EXCEPTION_DISPOSITION _except_handler4_common(
    IN PUINT_PTR CookiePointer,
    IN PCOOKIE_CHECK CookieCheckFunction,
    IN PEXCEPTION_RECORD ExceptionRecord,
    IN PEXCEPTION_REGISTRATION_RECORD EstablisherFrame,
    IN OUT PCONTEXT ContextRecord,
    IN OUT PVOID DispatcherContext
)
{
    EXCEPTION_POINTERS ExceptionInfo = { ExceptionRecord, ContextRecord };
    int result = ExceptionHandler(&ExceptionInfo);
    if (result == EXCEPTION_CONTINUE_EXECUTION) {
        return ExceptionContinueExecution;
    }
    if (result == EXCEPTION_CONTINUE_SEARCH) {
        return ExceptionContinueSearch;
    }
    return ExceptionCollidedUnwind;
}

```

3. Add (the name of the function) `_except_handler4_common` after `_except_handler3` in

`\base\crts\libw32\lib\dll\obj\i386\ntcrt.def`

4. In `\base\crts\libw32\lib\dll40\crt40.def`, add

```
_except_handler4_common = msvcrt._except_handler4_common PRIVATE
```

after

```
_except_handler3 = msvcrt._except_handler3 PRIVATE
```

5. Add the following `dirs` file to `\base\crts\crtw32\extras`

```
DIRS = \
    dll \
    dll_dbg
```

6. Add the following `lsources` file to `\base\crts\crtw32\extras`

```
CURDIR = extras

OBJSDIR=\
    $(OBJDIR)\chandler4.obj
```

7. Add the following `sources.nt` file to `\base\crts\crtw32\extras`

```
MAJORCOMP=crt
MINORCOMP=extras

TARGETNAME=extras

INCLUDES=..\..\h

!INCLUDE ..\..\crt32.nt

C_DEFINES = $(C_DEFINES) -DWINHEAP=1
```

8. Add the following `sources` file to `\base\crts\crtw32\extras\dll`

```
CRTLIBTYPE=DLL
!include ..\sources.nt

INCLUDES=..\..\..\..\..\extras\include\

SOURCES= \
    ..\chandler4.cpp \
    ..\gs_cookie.c \
    ..\gs_support.c \
    ..\gs_report.c
```

9. Copy `makefile` from `\base\crts\crtw32\dllstuff\dll` to `\base\crts\crtw32\extras\dll`

10. Copy `sources` from `\base\crts\crtw32\dllstuff\dll_dbg` to `\base\crts\crtw32\extras\dll_dbg`

11. Copy `makefile` from `\base\crts\crtw32\dllstuff\dll_dbg` to `\base\crts\crtw32\extras\dll_dbg`

12. Add `extras` at the end of the `DIRS` macro in `\base\crts\crtw32\dirs` - make sure that the line above it is terminated with `\`

13. Add `..\..\..\crtw32\extras\dll$(DOBJS)\$0\extras.lib` to the `LINKLIBS` and `OBJLIBFILES` macros in

`\base\crts\libw32\lib\sources.nt` - make sure that the line above it is terminated with `\`

14. In `\base\crts\makefile.inc`, add

```
$(OBJDIR)\extras.lib: $(EXTRA_OBJECTS)
    $(LIB) -out:$@ @<<$*.rsp
$(EXTRA_OBJECTS: =^
)
<<keep
```

15. In `\base\crts\makefile.inc`, after

```
!if "$(TARGET_CPU)" == "i386"
```

add

```
$(OBJDIR)\extras.lib: $(PREOBJDIR)\extras.lib
    copy $(PREOBJDIR)\extras.lib $@
```

16. In `\base\crts\makefile.inc`, after

```
$(OBJDIR)\time.lib: $(TIME_OBJECTS)
    $(LIB) -out:$@ @<<$*.rsp
$(TIME_OBJECTS: =^
)
<<keep
```

add

```
$(OBJDIR)\extras.lib: $(EXTRA_OBJECTS)
    $(LIB) -out:$@ @<<$*.rsp
$(EXTRA_OBJECTS: =^
)
<<keep
```

17. Run `bcz` in `\base\crt`
18. Overwrite the created `\base\crt\libw32\lib\dll\obj\i386\MSVCRT.DLL` in the `C:\WINDOWS\SYSTEM32` folder of your Windows XP installation