Anti-Debugging With Exceptions

John Leitch

8/29/2011

john@autosectools.com

http://www.autosectools.com/

Introduction

Several techniques for detecting exception swallowing debuggers have been documented. The concept is simple: by design, debuggers handle certain kinds of exceptions. If such an exception is wrapped in a try block, the exception handle is only executed if a debugger is not attached. Hence it can be inferred that a debugger is attached whenever the exception block is not executed.

One Byte Interrupt 3 (0xCC)

Among the most obvious of exceptions that a debugger would handle is the single byte breakpoint, generated by executing interrupt 3.

```
BOOL IsDebuggerPresent_Int3()
{
	__try
	{
		__asm int 3
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

Two Byte Interrupt 3 (0xCD 0x03)

An alternative two byte representation of interrupt 3 can be inserted by using the Visual C++ inline assembler __emit pseudoinstruction. Of all the debuggers tested only OllyDbg consumed the generated exception.

```
BOOL IsDebuggerPresent_Int3_2Byte()
{
	__try
	{
		__asm
		{
			__emit 0xCD
			__emit 0x03
		}
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

Interrupt 0x2C

Interrupt 0x2C raises a debug assertion exception. This exception is consumed by WinDbg, but none of the other debuggers tested handled it. The instruction can be generated using the __int2c intrinsic. This interrupt only works on Vista or later. Because of this, the windows version is dynamically checked, and if it is less than six false is returned.

```
BOOL IsDebuggerPresent_Int2c()
{
	OSVERSIONINFO osvi;

	ZeroMemory(&osvi, sizeof(OSVERSIONINFO));
	osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);

	GetVersionEx(&osvi);

	if (osvi.dwMajorVersion < 6)
		return FALSE;

	__try
	{
		__int2c();
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

Interrupt 0x2D

When interrupt 0x2D is executed Windows raises a breakpoint exception.

```
BOOL IsDebuggerPresent_Int2d()
{
	__try
	{
		__asm int 0x2d
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

ICEBP (0xF1)

ICEBP is an undocumented instruction that serves as a single byte interrupt 1, generating a single step exception.

```
BOOL IsDebuggerPresent_IceBp()
{
	__try
	{
		__asm __emit 0xF1
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

The Trap Flag

Bit number 8 of the EFLAGS register is the trap flag. When set, a single step exception is generated.

```
BOOL IsDebuggerPresent_TrapFlag()
{
	__try
	{
		__asm
		{
			pushfd
			or word ptr[esp], 0x100
			popfd
			nop
		}
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}
```

RaiseException

Several different types of exceptions generated with the RaiseException function are swallowed by debuggers. OutputDebugString, a function commonly used to detect an attached debugger, relies on RaiseException. The effective exception codes are as follows:

STATUS_BREAKPOINT	(0x80000003)
STATUS_SINGLE_STEP	(0x80000004)
DBG_PRINTEXCEPTION_C	(0x40010006)
DBG_RIPEXCEPTION	(0x40010007)
DBG_CONTROL_C	(0x40010005)
DBG_CONTROL_BREAK	(0x40010008)
DBG_COMMAND_EXCEPTION	(0x40010009)

ASSERTION_FAILURE	(0xC0000420)
STATUS_GUARD_PAGE_VIOLATION	(0x80000001)
SEGMENT_NOTIFICATION	(0x40000005)
EXCEPTION_WX86_SINGLE_STEP	(0x4000001E)
EXCEPTION_WX86_BREAKPOINT	(0x4000001F)

```
BOOL TestExceptionCode(DWORD dwCode)
{
	__try
	{
		RaiseException(dwCode, 0, 0, 0);
	}
	__except(1)
	{
		return FALSE;
	}

	return TRUE;
}

BOOL IsDebuggerPresent_RipException()
{
	return TestExceptionCode(DBG_RIPEXCEPTION);
}
```

Compatibility Chart

	VS2008 SP1	VS2010 SP1	WinDbg 6.12.1.633	OllyDbg 2.01A4	Ida 5.0
Interrupt 3 (1 Byte)	X	X	X	X	
Interrupt 3 (2 Byte)				X	
Interrupt 0x2C (Vista and later)			X		
Interrupt 0x2D	X	X	X	X	
ICEBP (0xF1)	X	X	X	X	
Trap Flag			X	X	
RaiseException STATUS_BREAKPOINT	X	X		X	
RaiseException STATUS_SINGLE_STEP	X	X	X	X	
RaiseException DBG_PRINTEXCEPTION_C	X	X	X	X	X
RaiseException DBG_RIPEXCEPTION	X	X	X	X	X
RaiseException DBG_CONTROL_C			X	X	
RaiseException DBG_CONTROL_BREAK			X	X	
RaiseException DBG_COMMAND_EXCEPTION			X	X	
RaiseException ASSERTION_FAILURE			X	X	
RaiseException STATUS_GUARD_PAGE_VIOLATION				X	
RaiseException SEGMENT_NOTIFICATION				X	
RaiseException EXCEPTION_WX86_SINGLE_STEP			X	X	
RaiseException EXCEPTION_WX86_BREAKPOINT				X	

Resources

- Anti-unpacker tricks - part two
  http://www.virusbtn.com/virusbulletin/archive/2009/01/vb200901-anti-unpacker-tricks
- Anti-unpacker tricks - part three
  http://www.virusbtn.com/virusbulletin/archive/2009/02/vb200902-anti-unpacker-tricks
- Windows Anti-Debug Reference
  http://www.symantec.com/connect/articles/windows-anti-debug-reference

- Undocumented OpCodes: ICEBP
  http://www.rcollins.org/secrets/opcodes/ICEBP.html
- EFLAGS Individual Bit Flags
  http://www.c-jump.com/CIS77/ASM/Instructions/I77_0070_eflags_bits.htm
- __int2c
  http://msdn.microsoft.com/en-us/library/5d6627x5.aspx
- Reversing: Secrets of Reverse Engineering
  http://www.amazon.com/Reversing-Secrets-Engineering-Eldad-Eilam/dp/0764574817
- The _emit Pseudoinstruction
  http://msdn.microsoft.com/en-us/library/1b80826t(v=vs.80).aspx
- NTSTATUS values
  http://msdn.microsoft.com/en-us/library/cc704588(v=prot.10).aspx