

Forum:[Hack Forums](#)

Thread: [Dynamic API call + Anti usermode hooking](#)

d3c0mpil3r - November 6, 2013, 1:46 am

Hey guys,

so I had implemented a simple method to bypass some usermode hooks in .NET but then it adds to your application's dependency, as it was based on a DLL.

More info here: Spoiler [\[Click to View\]https://hackforums.net/showthread.php?tid=3100410](https://hackforums.net/showthread.php?tid=3100410)

So, I re-wrote the code. Now, the (position-independent) code is copied to unmanaged memory and then executed.

This will bypass some of the common hooking techniques (IAT, hotpatching, etc.)

Here it is (AntiHook.vb):

Code Public Class AntiHook

#Region "WinAPI"

' d3c0mpil3r

```
<DllImport("kernel32.dll", CharSet:=CharSet.Auto)> _  
Public Shared Function GetModuleHandle(ByVal lpModuleName As String) As IntPtr  
End Function
```

```
<DllImport("kernel32.dll", SetLastError:=True, CharSet:=CharSet.Ansi, ExactSpelling:=True)> _  
Private Shared Function GetProcAddress(ByVal hModule As IntPtr, ByVal procName As String)  
As IntPtr  
End Function
```

```
<DllImport("kernel32.dll", SetLastError:=True)> _  
Public Shared Function WriteProcessMemory(ByVal hProcess As IntPtr, ByVal lpBaseAddress As  
IntPtr, ByVal lpBuffer As Byte(), ByVal nSize As UInt32, <Out()> ByRef lpNumberOfBytesWritten As  
IntPtr) As Boolean  
End Function
```

```
<DllImport("kernel32.dll", SetLastError:=True)> _  
Private Shared Function VirtualAlloc(lpAddress As IntPtr, dwSize As UIntPtr, flAllocationType As  
UInt32, flProtect As UInt32) As IntPtr  
End Function
```

```
<DllImport("kernel32.dll", SetLastError:=True)> _  
Private Shared Function VirtualFree(lpAddress As IntPtr, dwSize As UInt32, dwFreeType As  
UInteger) As Boolean  
End Function
```

#End Region

```
<UnmanagedFunctionPointer(CallingConvention.StdCall, SetLastError:=True)> _  
Public Delegate Function Run(ByVal lpAddress As UInt32, ByVal args As UInt32(), ByVal count
```

As Integer) As UInt32

Private Shared ReadOnly Code As Byte()

Shared Sub New()

```
Code = New Byte() {&H55, &H8B, &HEC, &H83, &HEC, &HC, _  
&H53, &H57, &HC7, &H45, &HF8, &H0, _  
&H0, &H0, &H0, &H83, &H7D, &H8, _  
&H0, &H75, &H4, &H33, &HC0, &HEB, _  
&H49, &H8B, &H45, &H10, &H83, &HE8, _  
&H1, &H89, &H45, &HFC, &HEB, &H9, _  
&H8B, &H4D, &HFC, &H83, &HE9, &H1, _  
&H89, &H4D, &HFC, &H83, &H7D, &HFC, _  
&H0, &H7C, &H12, &H8B, &H55, &HFC, _  
&H8B, &H45, &HC, &H8B, &HC, &H90, _  
&H89, &H4D, &HF4, &H8B, &H45, &HF4, _  
&H50, &HEB, &HDF, &H8B, &H45, &H8, _  
&HE8, &H0, &H0, &H0, &H0, &H5B, _  
&H83, &HC3, &HF, &H53, &H8B, &HFF, _  
&H55, &H8B, &HEC, &H83, &HC0, &H5, _  
&HFF, &HE0, &H89, &H45, &HF8, &H8B, _  
&H45, &HF8, &H5F, &H5B, &H8B, &HE5, _  
&H5D, &HC2, &HC, &H0}
```

End Sub

Public Shared Function Execute(ByVal dllName As String, ByVal funcName As String, ByVal
args() As UInt32) As UInt32

Dim retValue As UInt32 = 0

Dim funcAddr As IntPtr = GetProcAddress(GetModuleHandle(dllName), funcName)

If funcAddr = IntPtr.Zero Then

Return 0

End If

Dim memAddr As IntPtr = VirtualAlloc(IntPtr.Zero, 1000, &H1000 Or &H2000, &H40)

If memAddr = IntPtr.Zero Then

Return 0

End If

Marshal.Copy(Code, 0, memAddr, Code.Length)

Try

Dim r As Run = Marshal.GetDelegateForFunctionPointer(memAddr, GetType(Run))

retValue = r(funcAddr.ToInt32(), args, args.Length)

Catch ex As Exception

MessageBox.Show(ex.Message)

Return 0

Finally

VirtualFree(memAddr, 0, &H8000)

End Try

```
Return retValue  
End Function
```

```
End Class
```

Usage:

```
Code Dim msgAddr As IntPtr = Marshal.StringToHGlobalAnsi("Hello")  
Dim titleAddr As IntPtr = Marshal.StringToHGlobalAnsi("Title")
```

```
Dim args As UInt32() = {0, msgAddr, titleAddr, MessageBoxButtons.YesNo}
```

```
AntiHook.Execute("user32.dll", "MessageBoxA", args)
```

```
Marshal.FreeHGlobal(msgAddr)  
Marshal.FreeHGlobal(titleAddr)  
Enjoy!
```

0xBB - November 6, 2013, 3:19 am

Wow ! I've been working on something similar and I was stuck when it came to pass the other argument stacks. Good work !

frontenhack - November 6, 2013, 5:15 am

I've worked on something similar but I was loading net framework in a process with a shellcode for doing hooks :D

Only3 - November 6, 2013, 12:50 pm

did u test it on 64bit from ur uac thread probably u using 32 so this1 have problem with 64 too

Edit - ... <http://research.microsoft.com/en-us/projects/detours/>

d3c0mpil3r - November 7, 2013, 3:20 am

(Nov 6, 2013 01:50 PM)Only3 Wrote: [\[CITE\]](#) did u test it on 64bit from ur uac thread probably u using 32 so this1 have problem with 64 too

Edit - ... <http://research.microsoft.com/en-us/projects/detours/>

I haven't tested this on 64-bit. If someone could test this, that would be great!

0xBB - November 7, 2013, 5:04 am

(Nov 6, 2013 01:50 PM)Only3 Wrote: [\[CITE\]](#) did u test it on 64bit from ur uac thread probably u using 32 so this1 have problem with 64 too

Edit - ... <http://research.microsoft.com/en-us/projects/detours/>

Quit posting useless stuff man seriously. Why use a third lib when you can do it without detour ?

@OP , it work fine on x64 bits. ?

frontenhack - November 7, 2013, 5:17 am

It works if you compile x86 and not anycpu! :D

Sc?rf?c3_ - November 10, 2013, 4:23 am

Nice one d3c0mpil3r.... works fine, really appreciated this.