

Forum:[Hack Forums](#)

Thread: [\[VB.NET\] Very simple "Anti dll-injection"](#)

SIM-478284 - July 29, 2014, 10:44 am

Hello everyone. It's been a while since I shared something, even something little, so I decided I could share this quick snippet I came up with.

The code:

Spoiler ([Click to View](#))

```
Code Declare Function GetProcAddress Lib "kernel32" (ByVal ModuleHandle As Integer, ByVal ProcName As String) As Integer
    Declare Function GetModuleHandle Lib "kernel32" Alias "GetModuleHandleA" (ByVal lpModuleName As String) As IntPtr
    Declare Function WriteProcessMemory Lib "kernel32" (ByVal hProcess As IntPtr, ByVal lpBaseAddress As IntPtr, ByVal lpBuffer As Byte(), ByVal nSize As UInteger, ByRef lpNumberOfBytesWritten As UInt32) As Boolean

    Public Sub ScrewLoadLibrary()
        Dim LoadLibraryA As IntPtr = GetProcAddress(GetModuleHandle("kernel32"), "LoadLibraryA")
        Dim LoadLibraryW As IntPtr = GetProcAddress(GetModuleHandle("kernel32"), "LoadLibraryW")
        If LoadLibraryA <> IntPtr.Zero Then WriteProcessMemory(Process.GetCurrentProcess.Handle, LoadLibraryA, New Byte() {&HC2, &H4, &H0, &H90}, 4, 0)
        If LoadLibraryW <> IntPtr.Zero Then WriteProcessMemory(Process.GetCurrentProcess.Handle, LoadLibraryW, New Byte() {&HC2, &H4, &H0, &H90}, 4, 0)
    End Sub
```

This, is what either of LoadLibrary entry points look like before execution of the code:

Spoiler ([Click to View](#))

This is LoadLibrary entry point afterwards:

Spoiler ([Click to View](#))

The first statement of LoadLibrary was just replaced by a return statement, causing any dll injector using LoadLibrary to fail (because when LoadLibrary is being called, it just returns without loading the actual dll into the process).

All you have to do is to call "ScrewLoadLibrary()".

Hopefully somebody finds this useful.

Ps. This is easy to reverse though.

Madara-Uchiha - July 29, 2014, 10:52 am

Yo, buddy S4E posted some nice stuff there :)
Very useful!

RKYDA - July 29, 2014, 11:15 am

Hello, thank you for this.

With option strict:

```
Dim LoadLibraryA As IntPtr = CType(GetProcAddress(CInt(GetModuleHandle("kernel32")),  
"LoadLibraryA"), IntPtr)  
Dim LoadLibraryW As IntPtr = CType(GetProcAddress(CInt(GetModuleHandle("kernel32")),  
"LoadLibraryW"), IntPtr)
```

0xBB - July 29, 2014, 11:23 am

Interesting, you should redirect the loadlib arg to a ptr and check the library loaded. (I mean, what if my application use Loadlib to do stuff ?) and ret/jmp only if it doesn't match a list. Even so, I guess it could be bypassed easily D; !! Anyway nice share OP !

Only3 - July 29, 2014, 11:23 am

(Jul 29, 2014 11:15 AM)D[a]rk[y] Wrote: [\[CITE\]](#) Hello, thank you for this.

With option strict:

```
Dim LoadLibraryA As IntPtr = CType(GetProcAddress(CInt(GetModuleHandle("kernel32")),  
"LoadLibraryA"), IntPtr)  
Dim LoadLibraryW As IntPtr = CType(GetProcAddress(CInt(GetModuleHandle("kernel32")),  
"LoadLibraryW"), IntPtr)  
he use the old vb api  
change \ use  
Declare Function GetProcAddress Lib "kernel32" (ByVal ModuleHandle As IntPtr, ByVal ProcName  
As String) As IntPtr
```

d3c0mpil3r - July 29, 2014, 11:27 am

This implementation is flawed. Here's why:

You're screwing the LoadLibrary function even for the application itself. What if the app tries to load a dll via LoadLibrary at some point?

A parameter is passed to the function, so it overwritten instruction should be RET 4 instead (4 bytes for the pointer). Your current implementation will mess up the stack and crash the app.

iStalker - July 29, 2014, 11:38 am

(Jul 29, 2014 11:23 AM)Only3 Wrote: [\[CITE\]](#) he use the old vb api

change \ use

```
Declare Function GetProcAddress Lib "kernel32" (ByVal ModuleHandle As IntPtr, ByVal ProcName  
As String) As IntPtr  
'Declare' is old too...
```

PHP Code

```
<?php  
<DllImport("kernel32", CharSet:=CharSet.Auto, SetLastError:=True)>  
Private Shared Function GetProcAddress(  
    ByVal ModuleHandle As IntPtr,  
    ByVal ProcName As String) As IntPtr  
End Function
```

Only3 - July 29, 2014, 11:47 am

(Jul 29, 2014 11:27 AM)d3c0mpil3r Wrote: [\[CITE\]](#) This implementation is flawed. Here's why:
You're screwing the LoadLibrary function even for the application itself. What if the app tries to load a dll via LoadLibrary at some point?

A parameter is passed to the function, so it overwritten instruction should be RET 4 instead (4 bytes for the pointer). Your current implementation will mess up the stack and crash the app.

? simple if check will fix it ?

(Jul 29, 2014 11:38 AM)iStalker Wrote: [\[CITE\]](#) 'Declare' is old too...

PHP Code

```
<?php
<DllImport("kernel32", CharSet:=CharSet.Auto, SetLastError:=True)>
    Private Shared Function GetProcAddress(
        ByVal ModuleHandle As IntPtr,
        ByVal ProcName As String) As IntPtr
    End Function
```

its obvious dont u think i even write it in my reply ?

SIM-478284 - July 29, 2014, 11:55 am

(Jul 29, 2014 11:27 AM)d3c0mpil3r Wrote: [\[CITE\]](#) This implementation is flawed. Here's why:
You're screwing the LoadLibrary function even for the application itself. What if the app tries to load a dll via LoadLibrary at some point?

A parameter is passed to the function, so it overwritten instruction should be RET 4 instead (4 bytes for the pointer). Your current implementation will mess up the stack and crash the app.

"You're screwing the LoadLibrary function even for the application itself."

Well, that was obviously point of my code. If the application is using LoadLibrary it's supposed to use it BEFORE screwing up the entry point to make sure only those dll files that are necessary are loaded, and there's no chance to use CreateRemoteThread and inject a dll into the process.

"A parameter is passed to the function, so it overwritten instruction should be RET 4 instead (4 bytes for the pointer). Your current implementation will mess up the stack and crash the app."

Really? My bad then. Thanks for pointing that out. Anyhow, the application does not crash for me when I try to inject a dll file (but neither does the dll get loaded), so how's that then?

Anyhow, the code should be fixed now on that matter.

(Jul 29, 2014 11:23 AM)ibennz Wrote: [\[CITE\]](#) Interesting, you should redirect the loadlib arg to a ptr and check the library loaded. (I mean, what if my application use Loadlib to do stuff ?) and ret/jmp only if it doesn't match a list. Even so, I guess it could be bypassed easily D; !! Anyway nice share OP !

I see what you mean. :P Well, my code works in case you have loaded all the necessary libraries before screwing the entry point.

d3c0mpil3r - July 29, 2014, 12:17 pm

About the 2nd point, here's a pseudo code:

```
Code push <address of string representing the path>
call LoadLibrary
```

Now inside the function, you're just returning. What about that pointer which is still lying on the top of the stack? That's where the problem is. You need to clean up the stack your self. So, replace RET with RET 4. You're lucky it didn't crash your app.

Although your method will protect from trivial dll injection attacks, the approach is still flawed for real time applications. They will most likely use LoadLibrary at some point to load dlls at runtime. You're actually limiting the application itself by doing this. Also, one can still use CreateRemoteThread to inject dlls or even inject the code directly, as this implementation is easy to bypass.

You should really consider blocking the creation of new suspicious threads instead of blocking random LoadLibrary calls.

Dispersia - July 29, 2014, 1:14 pm

(Jul 29, 2014 11:23 AM)ibennz Wrote: [\[CITE\]](#) Interesting, you should redirect the loadlib arg to a ptr and check the library loaded. (I mean, what if my application use Loadlib to do stuff ?) and ret/jmp only if it doesn't match a list. Even so, I guess it could be bypassed easily D; !! Anyway nice share OP !

You could also simply just write the instructions back before injecting.

SIM-1456245 - July 29, 2014, 5:38 pm

You got some great shit right there bro, good job. But as already mentioned, you are actually blocking yourself from LoadLibrary. But that would be very rare situation, so you will be good.

DeMoNi - July 29, 2014, 7:13 pm

(Jul 29, 2014 01:14 PM)Dispersia Wrote: [\[CITE\]](#) You could also simply just write the instructions back before injecting.

+1, compare the function opcodes, and change them back if they are modified.

webinject - July 29, 2014, 7:29 pm

(Jul 29, 2014 11:23 AM)ibennz Wrote: [\[CITE\]](#) Interesting, you should redirect the loadlib arg to a ptr and check the library loaded. (I mean, what if my application use Loadlib to do stuff ?) and ret/jmp only if it doesn't match a list. Even so, I guess it could be bypassed easily D; !! Anyway nice share OP !

'redirect the load lib arg to a ptr'

i loled a little inside the proper term for this is a hook. nothing you .NET coders would understand

op are you retarded? This will eventually crash once a module tries to import a reference and can't seem to register it into the PEB.

Also this doesn't patch injection at all. They can use ldrloadaddll, and several other functions. They can also manually map the dll then call CRT.

And again, many other ways.

0xBB - July 29, 2014, 7:39 pm

(Jul 29, 2014 07:29 PM)webinject Wrote: [\[CITE\]](#) 'redirect the load lib arg to a ptr'

i loled a little inside

op are you retarded? This will eventually crash once a module tries to import a reference and can't seem to register it into the PEB.

Also this doesn't patch injection at all. They can use `ldrload.dll`, and several other functions. They can also manually map the dll then call CRT.

And again, many other ways.

When will you stop coming back to HF kingroot? No one want your shitty grabber. You're the only retarded here Mr "how do i hide api callz in cpp ?".

webinject - July 29, 2014, 7:40 pm

(Jul 29, 2014 07:39 PM)ibennz Wrote: [\[CITE\]](#) When will you stop coming back to HF kingroot? No one want your shitty grabber. You're the only retarded here Mr "how do i hide api callz in cpp ?".i'm deliberately lost.. i'm not that said person ?

Flatsack - July 29, 2014, 8:10 pm

(Jul 29, 2014 10:44 AM)Scripter4ever Wrote: [\[CITE\]](#) Hello everyone. It's been a while since I shared something, even something little, so I decided I could share this quick snippet I came up with.

The code:

Spoiler [\(Click to View\)](#)

```
Code Declare Function GetProcAddress Lib "kernel32" (ByVal ModuleHandle As Integer, ByVal ProcName As String) As Integer
```

```
    Declare Function GetModuleHandle Lib "kernel32" Alias "GetModuleHandleA" (ByVal lpModuleName As String) As IntPtr
```

```
    Declare Function WriteProcessMemory Lib "kernel32" (ByVal hProcess As IntPtr, ByVal lpBaseAddress As IntPtr, ByVal lpBuffer As Byte(), ByVal nSize As UInteger, ByRef lpNumberOfBytesWritten As UInt32) As Boolean
```

```
    Public Sub ScrewLoadLibrary()  
        Dim LoadLibraryA As IntPtr = GetProcAddress(GetModuleHandle("kernel32"), "LoadLibraryA")  
        Dim LoadLibraryW As IntPtr = GetProcAddress(GetModuleHandle("kernel32"), "LoadLibraryW")  
        If LoadLibraryA <> IntPtr.Zero Then WriteProcessMemory(Process.GetCurrentProcess.Handle, LoadLibraryA, New Byte() {&HC2, &H4, &H0, &H90}, 4, 0)  
        If LoadLibraryW <> IntPtr.Zero Then WriteProcessMemory(Process.GetCurrentProcess.Handle, LoadLibraryW, New Byte() {&HC2, &H4, &H0, &H90}, 4, 0)  
    End Sub
```

This, is what either of LoadLibrary entry points look like before execution of the code:

Spoiler [\(Click to View\)](#)

This is LoadLibrary entry point afterwards:

Spoiler [\(Click to View\)](#)

The first statement of LoadLibrary was just replaced by a return statement, causing any dll injector using LoadLibrary to fail (because when LoadLibrary is being called, it just returns without loading the actual dll into the process).

All you have to do is to call "ScrewLoadLibrary()".

Hopefully somebody finds this useful.

Ps. This is easy to reverse though. Stop trying to do native jobs in a safe environment. Do you even know how the operating system works?

samuelrizzo - October 22, 2015, 5:36 am

Can you 'translate' this method for Delphi?

Thanks

Vrekt - October 22, 2015, 2:10 pm

(Oct 22, 2015 05:36 AM)samuelrizzo Wrote: [\[CITE\]](#) Can you 'translate' this method for Delphi?

Thanks

This is a year old thread.

But nice tutorial very useful.

qabeel1 - October 7, 2018, 6:44 am

How do I do a task check on a program if it is an injector or not ?

Vyxn - October 7, 2018, 10:42 pm

(Oct 7, 2018 06:44 AM)qabeel1 Wrote: [\[CITE\]](#) How do I do a task check on a program if it is an injector or not ?

Loop on active process, then do the checking for each process (there's a lot of method for checking injector).