

Forum:[Hack Forums](#)

Thread: [MenaRunPE, Injects all type of .NET exe's \(x86 -> x64, x64 -> x86 and so on\)](#)

Menalix - September 5, 2014, 6:22 am

My injector, which support all type of .NET Injections:

<http://www19.zippyshare.com/v/54392072/file.html>

It only works with .NET exe's, as I don't set the thread context, this is the .NET section anyways so fck Native exe's :D

I'm not gonna give out the source, but I want to explain myself in this thread how you can do it yourself.

I will refer to this source-code from AeonHack:

<https://hackforums.net/showthread.php?tid=4420101>

As it can easily be modified to support more than just x86 -> x86 injections and is very easy to follow, but leave in mind his offsets is in decimal and not hex, mostly PE Header analyzing tools shows offsets in Hex.

Remember to read this: <http://msdn.microsoft.com/en-us/gg463119.aspx>

If you are interested in a more detailed description of how the PE Header actually is constructed.

So let's start, what is the difference in a 64bit and 32bit PE when we want to do a RunPE Injection?

1: First is the CONTEXT structure we use to get the Thread's Context to receive the PEBAddress is different, there is CONTEXT and CONTEXT_AMD64, both structures can be found in the winnt header file, the CONTEXT_AMD64 is quite hard to convert though.

And we use GetThreadContext when injector is 32bit and target is 32bit or injector is 64bit and target is 64bit, and we then use Wow64GetThreadContext when injector is 64bit and target is 32bit.

As I've seen some talking nonsense about using Wow64GetThreadContext when 64bit -> 64bit because they don't know what Wow64 is all about, I thought it was worth mentioning the right API usage here.

2: When we get the PEB Address to find BaseAddress of our Target process.

PHP Code

```
<?php
```

```
If Not ReadProcessMemory(Pi.ProcessHandle, Ebx + 8, BaseAddress, 4, ReadWrite) Then  
Throw New Exception()
```

On a 32bit process the pointer address to BaseAddress will be: Ebx + 8

(Ebx is PEBAddress)

on a 64bit process the offset to BaseAddress is 10 and not 8. Keep that in mind when you later on has to overwrite the old BaseAddress of the PE and when you read with rpm as shown above code.

3: The PEBAddress on 64bit is of course 64bit sized (Int64), there for make sure to correct RPM and WPM API's to not look like this in AeonHacks source:

PHP Code

```
<?php
```

```
<DllImport("kernel32.dll", EntryPoint="ReadProcessMemory"), SuppressUnmanagedCodeSecurity
```

```
> _
```

```
Private Shared Function ReadProcessMemory( _
```

```
ByVal process As IntPtr, _
```

```

ByVal baseAddress As Integer, _
ByRef buffer As Integer, _
ByVal bufferSize As Integer, _
ByRef bytesRead As Integer) As Boolean
End Function

```

```

<DllImport("kernel32.dll", EntryPoint:="WriteProcessMemory"),
SuppressUnmanagedCodeSecurity> _
Private Shared Function WriteProcessMemory( _
ByVal process As IntPtr, _
ByVal baseAddress As Integer, _
ByVal buffer As Byte(), _
ByVal bufferSize As Integer, _
ByRef bytesWritten As Integer) As Boolean
End Function

```

As you clearly see these API's is wrong when checking:

[http://msdn.microsoft.com/en-us/library/windows/desktop/ms680553\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms680553(v=vs.85).aspx)
[http://msdn.microsoft.com/en-us/library/windows/desktop/ms681674\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681674(v=vs.85).aspx)

```

LPVOID =IntPtr
LPCVOID = IntPtr
SIZE_T = UInteger

```

4: The ImageBase value in the PE Header is stored 4 bytes ahead from the 32bit offset when the file is 64bit, which means the 64bit offset is 34 and not 30, starting of the PE Header's BaseAddress

5: We actually don't have to do so much else now, but here's some good tips:

Use IsWow64Process function when determining if the target is a 32bit process.

When you want to inject a payload of 32bit into a 64bit process, you'll have to rewrite the flags offset in the .NET header to 1 instead of 3, which lets the 32bit payload go into 64bit, by removing the "32bit required" flag.

Example:

PHP Code

```
<?php
```

```
If PayloadIs64 = False Then
```

```
    If dwNetDirFlags = &H3 Then
```

'To make sure we dont rewrite flags on a Payload which is already AnyCPU Compiled, it will only slow us down'

```
    Buffer.SetByte(payload, dwPEHeaderAddress + &H398, &H1)
```

'Replaces the .NET Header Flag on a 32bit compiled payload, to make it possible doing 32bit -> 64bit injection, without modifying the physical target exe on the drive.'

```
    End If
```

```
End If
```

Read the wMachine value in the PE Header to know whether the Payload is 64bit or not.

I heard many saying: Injecting 64bit to 32bit is impossible, well yes. But not in .NET because the code first is compiled into whether 32bit or 64bit with the JIT Compiler on runtime, so we can manipulate it into 32bit or 64bit like we want before the exe is executed, so awesome when injecting malware or using injections for legit purposes.

This tutorial is not quite detailed, but anyone who knows what a pointer is and the different

datatypes and have read the PE Header documentation should be able to follow up.
I just don't want to let c&p'ers come here and steal shit :)

taktaktak - September 5, 2014, 6:59 am

O_0 nice tut man can't wait to look into this!

TR.Gen - September 6, 2014, 1:44 am

Very nice, detailed explanation, learn a lot from this (y)

Menalix - September 6, 2014, 9:55 am

(Sep 6, 2014 01:44 AM)PULSAR150X Wrote: [\[CITE\]](#) Very nice, detailed explanation, learn a lot from this (y)Thank you, even I think many is in the boat of: WHAT THE HELL IS HE TALKING ABOUT :P

DragonHunter - September 6, 2014, 10:02 am

(Sep 5, 2014 06:22 AM)Menalix Wrote: [\[CITE\]](#) It only works with .NET exe's, as I don't set the thread context, this is the .NET section anyways so fck Native exe's :DIf it only works for .net why even bother using API's if you could just use the .Net Framework to handle the stuff :P
Shouldn't be so different from loading plugins
Assembly.Load anyone ? ?

frontenhack - September 6, 2014, 10:03 am

(Sep 6, 2014 10:02 AM)DragonHunter Wrote: [\[CITE\]](#) If it only works for .net why even bother using API's if you could just use the .Net Framework to handle the stuff :P
Shouldn't be so different from loading plugins
Assembly.Load anyone ? ?
You are right, but loading in another appdomain would be better ;)

Only3 - September 6, 2014, 12:48 pm

(Sep 6, 2014 10:03 AM)frontenhack Wrote: [\[CITE\]](#) You are right, but loading in another appdomain would be better ;)
yes so u can just unload it when u finish
? menalix i knew somthing like this will come
Convert to pdf the microsoft pe docx <https://mega.co.nz/#fm/HoFWgZDA>

intrinsic - September 6, 2014, 2:49 pm

Since when can you inject x86 code into a x64 environment?

Menalix - September 7, 2014, 3:35 am

(Sep 6, 2014 10:02 AM)DragonHunter Wrote: [\[CITE\]](#) If it only works for .net why even bother using API's if you could just use the .Net Framework to handle the stuff :P
Shouldn't be so different from loading plugins
Assembly.Load anyone ? ?Well because it was only to make an example on how this works, and if someone tried to reverse it, it would be kinda useless to their crypter.
My example uses NtGetProcessInformation instead of GetThreadContext, as I don't want to share CONTEXT_AMD64 it's a really baddy converting.
And that the cool about a RunPE is injecting other exe's and not loading into your own app domain.

(Sep 6, 2014 02:49 PM)intrinsic Wrote: [\[CITE\]](#) Since when can you inject x86 code into a x64 environment?Has always been possible in .NET

Because .NET files is compiled at runtime, so the code is first run into 64bit or 32bit at runtime, and it's pretty obvious seen with the AnyCPU compile option.

Did you think .NET could be compiled into both on the same time? And why does no native languages have that option.

intrinsic - September 7, 2014, 11:46 am

(Sep 7, 2014 03:35 AM)Menalix Wrote: [\[CITE\]](#) Well because it was only to make an example on how this works, and if someone tried to reverse it, it would be kinda useless to their crypter.

My example uses NtGetProcessInformation instead of GetThreadContext, as I don't want to share CONTEXT_AMD64 it's a really baddy converting.

And that the cool about a RunPE is injecting other exe's and not loading into your own app domain.

Has always been possible in .NET

Because .NET files is compiled at runtime, so the code is first run into 64bit or 32bit at runtime, and it's pretty obvious seen with the AnyCPU compile option.

Did you think .NET could be compiled into both on the same time? And why does no native languages have that option.

It's not possible with both assemblies being compiled as x64 and x86.

.NET assemblies that are compiled with constant architectures have constant registers and datatypes.

Good try though, really.

If this is only for .NET why not just use Assembly.Load?

Also there is no such thing as "NtGetProcessInformation"

NtQueryInformationProcess is what you're looking for. All you're doing is changing the imagebase.

<https://hackforums.net/showthread.php?tid=4424517>

Unturned - September 7, 2014, 11:49 am

I'm stupid enough to understand this tut, it allows inject or read teh source code inside?

Menalix - September 7, 2014, 11:59 am

(Sep 7, 2014 11:46 AM)intrinsic Wrote: [\[CITE\]](#) It's not possible with both assemblies being compiled as x64 and x86.

.NET assemblies that are compiled with constant architectures have constant registers and datatypes.

Good try though, really.

If this is only for .NET why not just use Assembly.Load?

Also there is no such thing as "NtGetProcessInformation"

NtQueryInformationProcess is what you're looking for. All you're doing is changing the imagebase.

<https://hackforums.net/showthread.php?tid=4424517>

Try my Injector yourself, and you'll be prooven wrong, there's a reason I uploaded it.

If you're to lazy look on my old video: <https://www.youtube.com/watch?v=k2pLF56gXXw>

Cause Assembly.Load only works for the calling application.

Yea I misspelled the API, good find (y)

Nope that's not the only thing I do, pretty much AeonHacks RunPE states everything I do and any other public RunPE.

Washi - September 7, 2014, 12:32 pm

Anyone that suggested to use Assembly.Load instead, hasn't taken obfuscated applications into account.

intrinsic - September 7, 2014, 12:57 pm

(Sep 7, 2014 11:59 AM)Menalix Wrote: [\[CITE\]](#) Try my Injector yourself, and you'll be proven wrong, there's a reason I uploaded it.

If you're too lazy to look at my old video: <https://www.youtube.com/watch?v=k2pLF56gXXw>

Cause Assembly.Load only works for the calling application.

Yea I misspelled the API, good find (y)

Nope that's not the only thing I do, pretty much AeonHacks RunPE states everything I do and any other public RunPE.

First of all, do you know what a codecave is?

Secondly, you don't even know the name of the API you're using. It's not a 'misspell'.

You can't inject x86->x64 without performing heavens gate.

(Sep 7, 2014 12:32 PM)TheUnknownProgrammer Wrote: [\[CITE\]](#) Anyone that suggested to use Assembly.Load instead, hasn't taken obfuscated applications into account.

Yeah because people are going to encrypt their already obfuscated file.

A packed executable should work just fine as long as there is nothing being dropped or relocated.

Menalix - September 7, 2014, 1:21 pm

(Sep 7, 2014 12:57 PM)intrinsic Wrote: [\[CITE\]](#) First of all, do you know what a codecave is?

Secondly, you don't even know the name of the API you're using. It's not a 'misspell'.

You can't inject x86->x64 without performing heavens gate.

Yeah because people are going to encrypt their already obfuscated file.

A packed executable should work just fine as long as there is nothing being dropped or relocated.

Yes i know what code cave is..... Which has nothing to do with this, with a RunPE we simply don't let the exe load finish, and overwrites it's data with ours, simply put.

It was a misspell idiot, I was answering from my iPhone when I was with my GF answering, and wrote "Get" instead of "Query", get over it dude, how the hell would even make my injector work if I declared a wrong API name.

Just try my injector if you don't believe me, how can it be so hard?

intrinsic - September 7, 2014, 1:23 pm

(Sep 7, 2014 01:21 PM)Menalix Wrote: [\[CITE\]](#) Yes i know what code cave is..... Which has nothing to do with this, with a RunPE we simply don't let the exe load finish, and overwrites it's data with ours, simply put.

It was a misspell idiot, I was answering from my iPhone when I was with my GF answering, and wrote "Get" instead of "Query", get over it dude, how the hell would even make my injector work if I declared a wrong API name.

Just try my injector if you don't believe me, how can it be so hard?

If you knew what a codecave was you wouldn't have brought up the problem for 'assembly.load being only available on the calling application' How does loadlibrary work? you utter and complete retard.

'with a runpe we simply don't let the exe load finish' You clearly HAVE no technical knowledge of as

to what a 'RUNPE' actually does. You can inject a file into a remote process whether or not it has been loaded. Despite this you'd have to rebuild the IAT.

You clearly don't know anything about the PE Format, nor the WINDOWS OS.

Also it's InformationProcess NOT ProcessInformation. You got the entire name wrong.

"CONTEXT_AMD64 it's a really baddy converting." What's there to convert? Are you incapable of implementing multiple structures and placing a check for the reference size?

Menalix - September 7, 2014, 2:41 pm

(Sep 7, 2014 01:23 PM)intrinsic Wrote: [\[CITE\]](#) If you knew what a codecave was you wouldn't have brought up the problem for 'assembly.load being only available on the calling application' How does loadlibrary work? you utter and complete retard.

'with a runpe we simply don't let the exe load finish' You clearly HAVE no technical knowledge of as to what a 'RUNPE' actually does. You can inject a file into a remote process whether or not it has been loaded. Despite this you'd have to rebuild the IAT.

You clearly don't know anything about the PE Format, nor the WINDOWS OS.

Also it's InformationProcess NOT ProcessInformation. You got the entire name wrong.

"CONTEXT_AMD64 it's a really baddy converting." What's there to convert? Are you incapable of implementing multiple structures and placing a check for the reference size? I never mentioned assembly.load anywhere, else than saying this is a RunPE, and that's what I wanted to show since so many call it impossible to inject x86 .NET into x64 .NET.

If I wanted to talk about Assembly.Load I would have done that.

Dumbass, of course I know that, but we're not talking about those type of injections, here we talk about the RunPE method, therefor we make sure by suspending it at start that we don't have to patch the import address table.

Then lets say that, I'm still the first to publish a RunPE which can inject a x64 .NET into a x86 .NET exe.

Sorry I didn't check MSDN up for you, didn't expect you were to lazy for that, grab a cola or something to power up.

For the average .NET coder on HF not many can convert this type of structure, I think there is like 20 at maximum in this section out of 200 who knows what they are doing:

Code typedef struct DECLSPEC_ALIGN(16) _CONTEXT {

```
//  
// Register parameter home addresses.  
//  
// N.B. These fields are for convience - they could be used to extend the  
// context record in the future.  
//
```

```
DWORD64 P1Home;  
DWORD64 P2Home;  
DWORD64 P3Home;
```

```
DWORD64 P4Home;  
DWORD64 P5Home;  
DWORD64 P6Home;
```

```
//  
// Control flags.  
//
```

```
DWORD ContextFlags;  
DWORD MxCsr;
```

```
//  
// Segment Registers and processor flags.  
//
```

```
WORD  SegCs;  
WORD  SegDs;  
WORD  SegEs;  
WORD  SegFs;  
WORD  SegGs;  
WORD  SegSs;  
DWORD EFlags;
```

```
//  
// Debug registers  
//
```

```
DWORD64 Dr0;  
DWORD64 Dr1;  
DWORD64 Dr2;  
DWORD64 Dr3;  
DWORD64 Dr6;  
DWORD64 Dr7;
```

```
//  
// Integer registers.  
//
```

```
DWORD64 Rax;  
DWORD64 Rcx;  
DWORD64 Rdx;  
DWORD64 Rbx;  
DWORD64 Rsp;  
DWORD64 Rbp;  
DWORD64 Rsi;  
DWORD64 Rdi;  
DWORD64 R8;  
DWORD64 R9;  
DWORD64 R10;  
DWORD64 R11;  
DWORD64 R12;  
DWORD64 R13;  
DWORD64 R14;
```

```

DWORD64 R15;

//
// Program counter.
//

DWORD64 Rip;

//
// Floating point state.
//

union {
    XMM_SAVE_AREA32 FltSave;
    struct {
        M128A Header[2];
        M128A Legacy[8];
        M128A Xmm0;
        M128A Xmm1;
        M128A Xmm2;
        M128A Xmm3;
        M128A Xmm4;
        M128A Xmm5;
        M128A Xmm6;
        M128A Xmm7;
        M128A Xmm8;
        M128A Xmm9;
        M128A Xmm10;
        M128A Xmm11;
        M128A Xmm12;
        M128A Xmm13;
        M128A Xmm14;
        M128A Xmm15;
    } DUMMYSTRUCTNAME;
} DUMMYUNIONNAME;

//
// Vector registers.
//

M128A VectorRegister[26];
DWORD64 VectorControl;

//
// Special debug control registers.
//

DWORD64 DebugControl;
DWORD64 LastBranchToRip;
DWORD64 LastBranchFromRip;
DWORD64 LastExceptionToRip;
DWORD64 LastExceptionFromRip;
} CONTEXT, *PCONTEXT;

```

Menalix - September 7, 2014, 2:51 pm

(Sep 7, 2014 02:45 PM)intrinsic Wrote: [\[CITE\]](#) You fucking said 'assembly.load is only available in the calling application'

Are you fucking retarded? WHAT TYPE OF INJECTION? I'm TALKING ABOUT THE 'DYNAMIC FORKING' you utter complete fucking retard.

I seriously can't argue with you anymore you don't even understand what i'm saying.

What does other HF Coders have to do with converting an unmanaged structure? Its you that is writing the shit runpe.

Go back to fucking copy pasting you utter skid. Where exactly does it state you can use it to load a .NET file into a remote app domain:

[http://msdn.microsoft.com/en-us/library/system.reflection.assembly.load\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.reflection.assembly.load(v=vs.110).aspx)

? I've never played with that function, so well I'm just purely looking on MSDN here.

If you think RunPE's shit then go somewhere else and complain about them, this was thread to learn others how to make 64bit -> 32bit and opposit, possible in .NET with the well known RunPE method as around 99% of all .NET crypters uses it.

Go back to school and learn to manner you language, you über bad mannered guy.

intrinsic - September 7, 2014, 2:53 pm

(Sep 7, 2014 02:51 PM)Menalix Wrote: [\[CITE\]](#) Where exactly does it state you can use it to load a .NET file into a remote app domain:

[http://msdn.microsoft.com/en-us/library/system.reflection.assembly.load\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.reflection.assembly.load(v=vs.110).aspx)

? I've never played with that function, so well I'm just purely looking on MSDN here.

If you think RunPE's shit then go somewhere else and complain about them, this was thread to learn others how to make 64bit -> 32bit and opposit, possible in .NET with the well known RunPE method as around 99% of all .NET crypters uses it.

Go back to school and learn to manner you language, you über bad mannered guy.

THATS THE POINT OF A CODE CAVE

YOU DELUSIONAL FUCKING RETARD

Menalix - September 7, 2014, 3:07 pm

(Sep 7, 2014 02:53 PM)intrinsic Wrote: [\[CITE\]](#) THATS THE POINT OF A CODE CAVE
YOU DELUSIONAL FUCKING RETARD

Uuh yes because modifying the IL code on the run is what we were discussing here.

Sorry that I haven't made the technic which suits your liking, I'm very sorry, now just stay away thanks :)

Menalix - October 12, 2014, 1:08 pm

Published source for this now at: <https://hackforums.net/showthread.php?tid=4487085>

Fixed AeonHacks small API errors, and made the API's dynamically loaded due to it being a habit.

whitemask - January 9, 2015, 3:32 pm

thank that is really greate :) :) ?

Vuno - January 10, 2015, 1:39 pm

Copy and pasted from the dev Aeon. Wow what a nub.

Disasm - January 10, 2015, 2:37 pm

This injection method is terrible and should never be used. The fact that you think this is a cross reference between x64 and x86 baffles me.

Considering .NET is executed by the CLR you can change the base architecture in the headers. This is not a true 'x86->x64' injection. When will .NET programmers start learning native programming for shit like this? Stop misusing the language for it's main purpose.

Menalix - January 11, 2015, 3:53 am

(Jan 10, 2015 03:37 PM)Disasm Wrote: [\[CITE\]](#) This injection method is terrible and should never be used. The fact that you think this is a cross reference between x64 and x86 baffles me.

Considering .NET is executed by the CLR you can change the base architecture in the headers. This is not a true 'x86->x64' injection. When will .NET programmers start learning native programming for shit like this? Stop misusing the language for it's main purpose.

Where does I indicate I thinks that?

Exactly and that is what this does, manipulating the flags in the PE header and CLR Header of the payload, nope this is not a true x86 to x64 injection as it is bytecode and doesn't have any too do with x64 asm opcodes and etc.

If you dont think I can code Native look at my inline hook tutorial with asm code injection. This is an old thread btw

(Jan 10, 2015 02:39 PM)Vuno Wrote: [\[CITE\]](#) Copy and pasted from the dev Aeon. Wow what a nub. Are you sad that I made a decompile of your stupid cheat, be glad I didnt bother cracking it too. How is this a copy-paste? This is a tutorial on improving it?