

Forum:[Hack Forums](#)

Thread: [ProjectorPE | Stable RunPE substitute for .NET assemblies! C# AND VB.NET!](#)

gigajew - April 29, 2018, 1:09 pm

Finally managed to stabilize my .NET RunPE after researching some. If an exception is thrown when you use it, you're probably running a .NET version lower than the assembly you're trying to run. For Quasar; try .NET 3.5 or .NET 4.0

Hyperhacktive - April 29, 2018, 1:20 pm

Thank you I'll test it soon ?

ShallowME - May 6, 2018, 7:10 am

I think you just added 10 more crypters to the market. GL!

ChainChain - May 7, 2018, 10:12 am

nice work bro , thnx for sharing

gigajew - May 7, 2018, 3:00 pm

(May 7, 2018 05:52 AM)MrSn0w Wrote: [\[CITE\]](#) Thanks for sharing, the only problem i have is, when i try to test it with Quasar (on 4.0) it just open and close itself over and over again, but nothing happens :/

Sounds like your AV is detecting it in memory

Washi - May 8, 2018, 8:27 am

Okay so I have a couple of questions about this snippet.

What is the reason behind the thread affinity and critical regions?

PHP Code

```
<?php
```

```
Thread.BeginThreadAffinity();
```

```
Thread.BeginCriticalRegion();
```

Why are we changing the apartment state based on the amount of parameters the entry point has?

PHP Code

```
<?php
```

```
if (parameters.Length > 0)
```

```
{  
    assemblyExecuteThread.SetApartmentState(ApartmentState.STA);
```

```
}  
else  
{  
    assemblyExecuteThread.SetApartmentState(ApartmentState.MTA);
```

```
}
```

And what is the purpose of clearing the first 32 bytes of the module?

PHP Code

<?php

```
RtlZeroMemory(Marshal.GetHINSTANCE(assembly.ManifestModule), 32);
```

Also not sure about the value of the unwrap function to be honest. Couldn't you just use one of the [Encoding](#) classes and use [GetBytes](#) instead?

Cyber Satan - May 12, 2018, 7:02 pm

I get this "A first chance exception of type 'System.Reflection.TargetInvocationException' occurred in mscorlib.dll"

Visual Basic 2010 Express

.Net 2.0 and .Net 4 (Tried both)

Immient Monitor

narc0tic - May 19, 2018, 8:15 am

Entrypoint.Invoke has been known for a long time.

Its very very detected at runtime and should not be used in a crypter.

Nice source though.

gigajew - May 25, 2018, 5:26 pm

I accidentally got the SetApartmentState in wrong order, if > 0 set it to MTA, otherwise set it to STA

ubbelol - May 26, 2018, 2:33 am

(May 25, 2018 05:26 PM)gigajew Wrote: [\[CITE\]](#) I accidentally got the SetApartmentState in wrong order, if > 0 set it to MTA, otherwise set it to STA

The number of parameters is irrelevant to what thread apartment state you should use. Read Washi's comment.

Ramboll - July 3, 2018, 6:55 am

another joke ??? get some errors too

Interallahu - July 4, 2018, 6:16 pm

Dude, your just adding more encouragement to use Assembly.Load Stuff, compared to a Injector or a Unmanaged alternative

Plus, Its hard to obfusacate APIs, Especially RtlZeroMemory, I suggest not using such an Anti-Dump.

AVs detect RtlZeroMemory alot (Common) and when it gets to VT, it probably just already got 66/66

At least try using something Like Exception Filters or Int2D, not to be rude and a Delegate for 1/2 method doesnt make much sense.. ?

SIM-3947811 - November 14, 2018, 7:04 pm

Works perfect for me. Idk why people says : IT'S DETECTED OMGGGG. It's looks like loadpe, very clean, no api calls. Thanks for share!

BTW, i test it now, hollows hunter and runpe hunter doesn't detect this ! VERY NICE !

Anonz8 - August 12, 2019, 6:25 pm

nice work bro , thnx for sharing :)

xHeaven - August 13, 2019, 2:26 am

(Nov 14, 2018 08:04 PM)hiddenkitty Wrote: [\[CITE\]](#) Works perfect for me. Idk why people says : IT'S DETECTED OMGGGG. It's looks like loadpe, very clean, no api calls. Thanks for share!

BTW, i test it now, hollows hunter and runpe hunter doesn't detect this ! VERY NICE !
You make no sense. This is nothing like a LoadPE, I wouldn't even comment the rest of your reply.

x64-re - October 24, 2020, 8:57 am

The code is fine, but this method will soon be detected by every AI out there (heuristics is getting replaced by AI). I'm going to send you a private message. I have some questions to ask, regarding your Github. It seems I can't, since I don't have 25 HQ replies yet, but I'll get there. I only make posts when it's necessary.

Elijah - October 24, 2020, 10:05 am

(Oct 24, 2020 08:57 AM)x64-re Wrote: [\[CITE\]](#) The code is fine, but this method will soon be detected by every AI out there (heuristics is getting replaced by AI). I'm going to send you a private message. I have some questions to ask, regarding your Github. It seems I can't, since I don't have 25 HQ replies yet, but I'll get there. I only make posts when it's necessary.
Can you share any good option regarding it?

x64-re - October 24, 2020, 10:13 am

(Oct 24, 2020 10:05 AM)Elijah Wrote: [\[CITE\]](#) Can you share any good option regarding it?
Obviously — but I'm not santa. I won't give out 0-days for free.

Soon enough I will sell my own crypter on here, in C++, and you can feel free to reverse it (good luck) and find out how it works. But the classic crypter method will be gone by 2022, at the very very latest.

Menalix - October 25, 2020, 2:14 am

(Jul 4, 2018 06:16 PM)Interallahu Wrote: [\[CITE\]](#) Dude, your just adding more encouragement to use Assembly.Load Stuff, compared to a Injector or a Unmanaged alternative

Plus, Its hard to obfusacate APIs, Especially RtlZeroMemory, I suggest not using such an Anti-Dump.

AVs detect RtlZeroMemory alot (Common) and when it gets to VT, it probably just already got 66/66

At least try using something Like Exception Filters or Int2D, not to be rude and a Delegate for 1/2 method doesnt make much sense.. ?

It's the opposite, when using .net functions instead of native api's directly, you'll have a harder time hiding the calls or what you do from AV's.

OP: Btw that patch I'm credited for, it's done wrong. I hope it's not my mistake lol, can't think of why I would do it like that.

JoshuaSmith - November 2, 2020, 5:29 pm

(Apr 29, 2018 01:09 PM)gigajew Wrote: [\[CITE\]](#) Finally managed to stabilize my .NET RunPE after researching some. If an exception is thrown when you use it, you're probably running a .NET version lower than the assembly you're trying to run. For Quasar; try .NET 3.5 or .NET 4.0

OFFTOPIC:

its very interesting how HF works, this code has no error handling barely follows, standards or the

naming conventions and its poorly written if you are referring to "by the book" its still a nice release and OP is a decent developer but if you have a blue name and post code you get people who rarely challenge vs someone with a white/green name.

You should re-code a bit of that as its can be simplified and restructured/cleaned but other than that nice release.

Menalix - November 3, 2020, 7:41 pm

(Nov 2, 2020 06:29 PM)JoshuaSmith Wrote: [\[CITE\]](#) OFFTOPIC:

its very interesting how HF works, this code has noerror handling barely follows, standards or the naming conventions and its poorly written if you are referring to "by the book" its still a nice release and OP is a decent developer but if you have a blue name and post code you get people who rarely challenge vs someone with a white/green name.

You should re-code a bit of that as its can be simplified and restructured/cleaned but other than that nice release.

Honestly it's not a very good release, for several of reasons:

1. His patch for Flags in .NET directory, calculates the address wrong, and it will high likely always mismatch, since .NET PE's .text section usually is at offset 0x200 and the flags is at offset 0x10 in the .NET directory, and the .NET directory is at beginning (offset 0x8 for Any/x86 and offset 0x0 for x64) of .text section

If the address happened to be right for the flags, he patches 1-byte, when the flags is an UInteger data-type, and therefor high likely wouldn't work either way. But there's no reason to patch it anyway obviously.

But the current "patch" will high likely do a 1-byte patch mid-way into the .text section, which will cause issues unless the .text section isn't bigger than 1mb, then you'll hit the .reloc section or some other section. You can also end with a memory access exception, if .net assembly is small enough for the byte patch to patch out of the memory-mapping.

2. No error handling, for example for the memory access exception which will occur on .NET assembly with less than 1mb in total size.

3. "BeginCriticalRegion" is useless

4. "BeginThreadAffinity" is useless

5. "EndCriticalRegion" even more useless, the thread is about to end anyway?

6. "EndThreadAffinity" just as useless, the thread is still about to end anyway?

7. SetApartmentState(ApartmentState.STA) doesn't have to be set for entrypoints with parameters...

8. SetApartmentState(ApartmentState.MTA) doesn't have to be set for entrypoints with no parameters...

9. Using GetCustomAttributes you can get the EntryPoint ApartmentState attribute and match is properly.

10. Uses RtlZeroMemory for no reason to erase 36-bytes of the assembly header. If it's for anti-dump, then I'm pretty sure most people know how to re-construct the dos header, only two variables is used there for loading PE's and one of the values is static (the chars: MZ), and the second (e_ifanew) is almost static, as it makes no sense to alter it, unless you decide to build a smaller custom dos_stub or entirely remove it, both for very odd reasons.

11. Not unloadable since it loads in the current AppDomain, doesn't make sense to load a full assembly in the current AppDomain when you aren't going to interact with it, when it comes with the cost of being unable to unload.

12. Using <SuppressUnmanagedCodeSecurity> when not needed

13. Unwrap function in VB variant, not functional, as he decided to only convert chars which represents a digit

14. Uses a delegate for no reason.

15. Has a variable for parameter for a nice parameter option in the initially run call, but decides to null it and not use it.

16. Calling it ProjectorPE is also misinforming, it should be called ProjectorNetAssembly.

17. Not setting stacksize of the thread to that of specified of assembly.

18. His initial comment to his own project: "Finally managed to stabilize my .NET RunPE after researching some"

Really doesn't fit with what we see, like not a single try catch, and not seperating the payload to its own AppDomain.

And it's not a "RunPE" It's just a Assembly.Load call, using the AppDomain.Load wrapper, calling the Assembly entrypoin in it's own thread.

This is rather a managed LoadPE and should be called LoadNetAssembly

I could honestly do this in half the amount of code, while stabilizing it with try catches and loading it in its own AppDomain.

Either way the code just looks like some dude try-hardening whatever, because he doesn't know a thing about what he is doing.

Seems like the only critical comments on this is also from knowledgeable devs like: @ubbelol and @Washi.