

Forum: [Hack Forums](#)

Thread: [Bypassme/Crackme/Unpackme - .NET With native layer protection + VMP](#)

fisica638 - October 31, 2023, 3:00 pm

I took my NETBuilderInjection and turned it into a Protector, but I don't know how secure it is. (For me, it's safe enough.)

So I made this little program, The objectives are:

[Unpackme] You should unpack it and get the cleanest code possible.

[Crackme] In the textbox where the HWID is displayed you must modify the text and write something else in it.

[Bypass] You will need to disable "Falcon Anticheat" from the executable.

In any of the objectives you must share the executable, to verify its work.

Difficulty : Unknown

Language Base : .NET

Platform : Windows - .NET Framework 4.6

OS Version : All

Packer / Protector : Improved NETBuilderInjection (in C) + VMP

Description :

NETBuilderInjection allows you to generate a Dll.NET assembly that can be easily injected into any process like any native dll. but in itself it is a packer. I made some improvements to it.

Download:

[UnPackMe.7z](#)

Screenshot :

Tentco - November 1, 2023, 8:07 pm

Alright hideaki or the AC dev

I will try it when I have time, I can smell usage of donut by the dropped Dll.

Exactly what is being used - <https://github.com/TheWover/donut/blob/master/DonutTest/Program.cs>

So, the VMP packed exe drops donut template which injects shc using VirtualAllocEx.

Hell it's the new vmp ver, they've added more crc and anti-dbg checks.

The shit AC is from <https://github.com/DestroyerDarkNess/Falcon-Anticheat>

Alright, so it's managed_method calling 'Load' from c++.

All that left is deobfuscating the Dll and extracting shc.

The icon, resources are taken from another dll, some sort of resource compression?

Anyway this is very insecure loader/dropper overall.

I will complete this when I have time.

Also the anti-vm is shit too, anyone can load unsigned driver of titanhide or hyperhide by exploiting Intel NIC iqvw64e.sys(Talking about VMP anti-vm check).

(Oct 31, 2023 03:00 PM)fisica638 Wrote: [\[CITE\]](#) I took my NETBuilderInjection and turned it into a Protector, but I don't know how secure it is. (For me, it's safe enough.)

So I made this little program, The objectives are:

[Unpackme] You should unpack it and get the cleanest code possible.

[Crackme] In the textbox where the HWID is displayed you must modify the text and write something else in it.

[Bypass] You will need to disable "Falcon Anticheat" from the executable.

In any of the objectives you must share the executable, to verify its work.

Difficulty : Unknown

Language Base : .NET

Platform : Windows - .NET Framework 4.6

OS Version : All

Packer / Protector : Improved NETBuilderInjection (in C) + VMP

Description :

NETBuilderInjection allows you to generate a Dll.NET assembly that can be easily injected into any process like any native dll. but in itself it is a packer. I made some improvements to it.

Download:

[UnPackMe.7z](#)

Screenshot :

Here, since the shellcode is PIC based no metadata is taken and I cannot convert it to CIL, this is the extracted shellcode injecting itself.

fisica638 - November 3, 2023, 4:41 pm

(Nov 1, 2023 08:07 PM)Tentco Wrote: [\[CITE\]](#) Alright hideaki or the AC dev

I will try it when I have time, I can smell usage of donut by the dropped Dll.

Exactly what is being used - <https://github.com/TheWover/donut/blob/master/DonutTest/Program.cs>

So, the VMP packed exe drops donut template which injects shc using VirtualAllocEx.

Hell it's the new vmp ver, they've added more crc and anti-dbg checks.

The shit AC is from <https://github.com/DestroyerDarkNess/Falcon-Anticheat>

Alright, so it's managed_method calling 'Load' from c++.

All that left is deobfuscating the Dll and extracting shc.

The icon, resources are taken from another dll, some sort of resource compression?

Anyway this is very insecure loader/dropper overall.
I will complete this when I have time.

Also the anti-vm is shit too, anyone can load unsigned driver of titanhide or hyperhide by exploiting Intel NIC iqvw64e.sys(Talking about VMP anti-vm check).

Here, since the shellcode is PIC based no metadata is taken and I cannot convert it to CIL, this is the extracted shellcode injecting itself.

Great bro, you're only halfway done. since you've only removed the native layer and the loader from the shellcode. You could have achieved the same thing, with less effort, if you had just Dumped the process using Extreme Dumper.

you got to the loader that loads the shellcode. Now you just need to get the original assembly and solve some of the objectives:

- 1) Get the cleanest binary. (that the original code is at least readable)
- 2) disable falcon anticheat.
- 3) edit the text where your HWID is displayed.

The native layer is insecure for two reasons:

1) Write the shellcode loader to disk. (I'm looking for a way to do it all from memory, maybe manual map? or reflective dll injection)

2) I haven't gotten a decent antidump.

However, you still have not achieved the original code or any objective.

Tentco - November 3, 2023, 6:22 pm

(Nov 3, 2023 04:41 PM)fisica638 Wrote: [\[CITE\]](#) Great bro, you're only halfway done. since you've only removed the native layer and the loader from the shellcode. You could have achieved the same thing, with less effort, if you had just Dumped the process using Extreme Dumper.

you got to the loader that loads the shellcode. Now you just need to get the original assembly and solve some of the objectives:

- 1) Get the cleanest binary. (that the original code is at least readable)
- 2) disable falcon anticheat.
- 3) edit the text where your HWID is displayed.

The native layer is insecure for two reasons:

1) Write the shellcode loader to disk. (I'm looking for a way to do it all from memory, maybe manual map? or reflective dll injection)

2) I haven't gotten a decent antidump.

However, you still have not achieved the original code or any objective.

Because you cannot decompile donut shellcode bud.

The image I attached is me debugging the independent shellcode using runsc.

fisica638 - November 4, 2023, 2:26 pm

(Nov 3, 2023 06:22 PM)Tentco Wrote: [\[CITE\]](#) Because you cannot decompile donut shellcode bud. The image I attached is me debugging the independent shellcode using runsc. In the image you attached, it appears that you are debugging the loader code (which is written in .net), however it is not the original code, the original binary does not touch the disk, it is loaded directly into memory.

To achieve the original code you need a good Memory Dumper. and then you need to remove the normal .net protections, which would be obfuscation, control flow, among other .net protections.

PD: You said my Falcon Anticheat sucks. :(?

Tentco - November 4, 2023, 7:27 pm

(Nov 4, 2023 02:26 PM)fisica638 Wrote: [\[CITE\]](#) In the image you attached, it appears that you are debugging the loader code (which is written in .net), however it is not the original code, the original binary does not touch the disk, it is loaded directly into memory.

To achieve the original code you need a good Memory Dumper. and then you need to remove the normal .net protections, which would be obfuscation, control flow, among other .net protections.

PD: You said my Falcon Anticheat sucks. :(?

Are you dumb? That's not .NET.

<https://github.com/edygert/runsc>

Your final donut template loader runs the donut shellcode which I extracted it cannot be converted into .NET, I wonder how many times I will have to say this.

When did I said it loads on disk, can't you see VirtualAlloc call in the image.

Improvements:

1. Improve your obfuscation.
 2. You can dump reflective injection, don't use it.
 3. Reprogram your AC in c or c++ with a driver to access low-level resources.
 4. VMP sucks.
 5. Even your process detection is based on Hwnd handle check.
 6. You're using IsDebuggerPresent() for anti-debug ... really?
 7. For a "decent" Anti-Dump set PAGE_GUARD in memory region using ZwProtectVirtualMemory, nullify NT and DOS headers at runtime(semi-polymorphic).
- There are more improvements I can list out but not here.

Watch this - <https://streamable.com/kgexpj>

The reason I can't complete this is because you cannot decompile/reverse donut shellcode, since no metadata is stored.

However I can load the native code in IDA or Ghidra, I don't have time to do this though due to work.

You straight up posted a PIC shellcode ldr without knowing whether it can be decompiled or not.

fisica638 - November 5, 2023, 5:26 am

The modified ExtremeDumper I have can dump the original assembly directly, leaving it without the native layer and without the loader. only to remove .NET protections. That's why I need a good

antidump.

Don't get me wrong, everything you said is right, however, you focused a lot on the native layer and the shellcode loader.

About antidebug, if I use IsDebuggerPresent and more, is there any antidebug that doesn't also use it?

<https://github.com/DestroyerDarkNess/DestroyerProtection/blob/master/DestroyerProtection/DestroyerCore/AntiDebug/MainDebug.vb>

The idea of this unpackme is to check the security of the packer, the native layer needs a lot of improvements, but even so you have not achieved the main objective, not even the original .NET assembly.... Maybe it's silly, but I achieved the original assembly. ?

Come on, let's not start with the insults, instead thank you for taking the time to analyze this.

Tentco - November 5, 2023, 5:46 am

(Nov 5, 2023 06:26 AM)fisica638 Wrote: [\[CITE\]](#) The modified ExtremeDumper I have can dump the original assembly directly, leaving it without the native layer and without the loader. only to remove .NET protections. That's why I need a good antidump.

Don't get me wrong, everything you said is right, however, you focused a lot on the native layer and the shellcode loader.

About antidebug, if I use IsDebuggerPresent and more, is there any antidebug that doesn't also use it?

<https://github.com/DestroyerDarkNess/DestroyerProtection/blob/master/DestroyerProtection/DestroyerCore/AntiDebug/MainDebug.vb>

The idea of this unpackme is to check the security of the packer, the native layer needs a lot of improvements, but even so you have not achieved the main objective, not even the original .NET assembly.... so maybe the fool is someone else. ?

I haven't called you fool but do you even know that donut uses the native part only? intermediate bytecode* and that you cannot convert it to MSIL neither you can unpack, reverse, modify it? Do you know that donut uses PIC?

The whole point is you've posted a challenge which is technically impossible and you don't know about it.

fisica638 - November 5, 2023, 5:57 am

When I have time, I will record a video of how I do it. However, you're right, I don't know native reversing, I only know .NET reversing.

Tentco - November 5, 2023, 5:59 am

(Nov 5, 2023 06:57 AM)fisica638 Wrote: [\[CITE\]](#) When I have time, I will record a video of how I do it. However, you're right, I don't know native reversing, I only know .NET reversing.

Buddy, I want to see how you will decompile PIC based shellcode into .NET.

fisica638 - November 5, 2023, 8:05 am

Only with Extreme Dumper. Now all that remains is to clean the assembly. (remove the .NET protections, I use my modification of confuserEx, LizProtect)

When working with packaged .NET assemblies, the first thing to do is dump the process.

Tentco - November 5, 2023, 8:07 am

(Nov 5, 2023 09:05 AM)fisica638 Wrote: [\[CITE\]](#) Only with Extreme Dumper. Now all that remains is to clean the assembly. (remove the .NET protections, I use my modification of confuserEx, LizProtect)

That's not the shellcode.

Decompile the shellcode, don't show modified exe which is not using shellcode lol.

fisica638 - November 5, 2023, 8:11 am

(Nov 5, 2023 09:07 AM)Tentco Wrote: [\[CITE\]](#) That's not the shellcode.

You will find the shell code in the ILoader.dll code, which was also dumped.

What I mean is that it is not necessary to do everything you did, just by dumping the process, you get the Shellcode Loader and the original protected assembly. There was no need to even remove VMP. I mean everything you did is unnecessary.

For this reason I need an antidump.

Tentco - November 5, 2023, 8:13 am

(Nov 5, 2023 09:11 AM)fisica638 Wrote: [\[CITE\]](#) You will find the shell code in the ILoader.dll code, which was also dumped.

Buddy you dropped unpackme.dll in dnspy in the vid and now you're showing me Iloader.dll. Please stop embarrassing yourself.

fisica638 - November 5, 2023, 8:24 am

(Nov 5, 2023 09:13 AM)Tentco Wrote: [\[CITE\]](#) Buddy you dropped unpackme.dll in dnspy in the vid and now you're showing me Iloader.dll.

Please stop embarrassing yourself.

watch the video again, I showed you the original assembly dump "HaxOfuscator_XXXX.dll " and the original "Unpackme.dll", and opened the resources so you can see that they are the same binaries.

I even underlined the directory in the explorer, before showing the original .exe without any protection. and compare the resources so you can see that they are the same binary.

You just have to accept that you couldn't. and you were trying to solve it from a crooked approach. Everyone knows that to revert to packaged PE.NET, you have to dump the process.

Tentco - November 5, 2023, 9:12 am

(Nov 5, 2023 09:24 AM)fisica638 Wrote: [\[CITE\]](#) watch the video again, I showed you the original assembly dump "HaxOfuscator_XXXX.dll " and the original "Unpackme.dll", and opened the

resources so you can see that they are the same binaries.

I even underlined the directory in the explorer, before showing the original .exe without any protection. and compare the resources so you can see that they are the same binary.

You just have to accept that you couldn't. and you were trying to solve it from a crooked approach. Everyone knows that to revert to packaged PE.NET, you have to dump the process.

You're funny you still couldn't show a video of decompiling shellcode to .NET, you're saying some stuff in convo and some other stuff here, grow up kid.

(Nov 5, 2023 09:24 AM)fisica638 Wrote: [\[CITE\]](#) watch the video again, I showed you the original assembly dump "HaxOfuscator_xxxx.dll " and the original "Unpackme.dll", and opened the resources so you can see that they are the same binaries.

I even underlined the directory in the explorer, before showing the original .exe without any protection. and compare the resources so you can see that they are the same binary.

You just have to accept that you couldn't. and you were trying to solve it from a crooked approach. Everyone knows that to revert to packaged PE.NET, you have to dump the process.

You in convo,

What is bro arguing on while agreeing to my point, I'm confused.

Holy, you're dropping the file which wasn't converted to shellcode on dnspy and comparing with the file you gave out as challenge(only the compressed resources container) totally people will not realize it.

In my first ever post on this thread I already talked about resource compression dll, go read again.

fisica638 - November 5, 2023, 11:50 am

The point, the unpackme has not been resolved, so I will leave this discussion pointless.

I'll just ask you one question: Can you solve it? Yes or no?

In case your answer; No.

That means I have achieved my goal, I can use it as a shield for .NET, I just need some improvements.

Technically, if the PC can understand and load the program, it can be reversed, even if it is very complicated... or at least that's what I think, am I wrong?