

Forum:[Hack Forums](#)

Thread: [Mono.Cecil -> Dnlib | Use Dnlib like Mono.Cecil](#)

Wumbology Expert - March 30, 2014, 5:43 pm

Prerequisites:

I saw a lot of people playing around with mono.cecil and using it as obfuscation, and nearly no one here seems to use the alternative, dnlib. The goal of this thread is to teach you all about dnlib and show you how to "convert" or "port" the mono.cecil code to the dnlib equivalent. Please do not flame the thread. I know I am new here, but I have been coding for a few years, but I am just now starting to get into malware and things like that.

Dnlib Download: <https://bitbucket.org/0xd4d/dnlib>

Mono.Cecil Download: <http://mono-project.com/Cecil>

If you want to download Mono.Cecil, I think you need to install it, then go to the Program Files folder for it and copy the Mono.Cecil.dll file to a location where you want to use it. For Dnlib, I think you need to download the source and compile it yourself.

Tutorial:

First off you want reference the Dnlib library:

1. Once you have created a project, double-click on "My Project" in your Solution Explorer tab.
2. Then find and click the reference tab.
3. There then should be a button in that tab that says "Add." Click it.
4. Find your Dnlib.dll file and open it.
5. You have now referenced that library

Next Up, you will want to import the Dnlib library to your code:

Next, I am assuming all of you will want to load the file you are trying to edit:

Mono.Cecil Code:

```
Code Dim aDef As AssemblyDefinition = AssemblyDefinition.ReadAssembly("PATH")
```

Dnlib Code:

```
Code Dim mdef As ModuleDefMD = ModuleDefMD.Load("PATH")
```

Next, you would want to loop through each class in the application. You also want to make sure that the Application has any class or namespaces:

Mono.Cecil Code:

```
Code If aDef.MainModule.HasTypes Then  
    For Each cls In aDef.MainModule.Types()
```

Next

```
End If
Dnlib Code:
Code If mdef.HasTypes Then
    For Each cls In mdef.Types()

        Next
    End If
```

Next, this is what you do in order to loop through each method in a class or namespace. You would put this inside the class loop. Once again, you also want to make sure that the class has methods:

```
Mono.Cecil Code:
Code If cls.HasMethods Then
    For Each mthd In cls.Methods()

        Next
    End If
Dnlib Code:
Code If cls.HasMethods() Then
    For Each mthd In cls.Methods()

        Next
    End If
```

Next, you may want to loop through each variable in a method, so here is what you would do inside of the method loop. And again, you need to make sure that your method has a body and has variables in the body:

```
Mono.Cecil Code:
Code If mthd.HasBody() And mthd.Body.HasVariables() Then
    For Each var In mthd.Body.Variables()

        Next
    End If
Dnlib Code:
Code If mthd.HasBody And mthd.Body.HasVariables Then
    For Each var In mthd.Body.Variables()

        Next
    End If
```

Renaming:

Renaming is basically just taking the loop variable and finding the name property and setting it to something else. I would also like to say that renaming classes, methods, and variables may cause your application to crash. You will need to make checks in your method to make sure you do not rename certain variables, methods, and classes.

```
Both Mono.Cecil Code and Dnlib Code
Code cls.Name = "BLAHBLAH"
mthd.Name = "BLAHBLAH"
var.Name = "BLAHBLAH"
```

Summation:

As you can see, the code is nearly exactly the same for method loops and variable loops, so porting it was pretty easy. These libraries are not exactly the same. There are differences in the way they handle the executable when the file is loaded and saved. You can do basically everything you can do in dnlib that you can do in mono.cecil.

Final Notes:

If you want to take renaming even further, you can make loops to loop through other aspects of code such as Fields, Properties, and Resources. You also have to take into account if the method is a constructor, or if the names are Special Names or not. I do not want to hand everyone the code, so I will not be giving everyone the exact code. I hope this helps you all, and if you want me to add something to this, please tell me.

ubbelol - March 31, 2014, 2:35 am

I don't really think this thread is necessary. Most of the code is really similar and easy to figure out. And it's better to just let people ask specific questions rather than trying to explain it all here. :P

Additionally if someone can use Mono.Cecil but not figure out how to use dnlib you're doing something wrong.

Wumbology Expert - March 31, 2014, 5:26 am

(Mar 31, 2014 02:35 AM)ubbelol Wrote: [\[CITE\]](#) I don't really think this thread is necessary. Most of the code is really similar and easy to figure out. And it's better to just let people ask specific questions rather than trying to explain it all here. :P

Additionally if someone can use Mono.Cecil but not figure out how to use dnlib you're doing something wrong.

Well, I guess if someone has a question, they can ask it. I know the code is really similar, but a lot of people also don't know about dnlib, so I guess this could be an introduction for them as well.

Dispersia - March 31, 2014, 8:27 am

I see nothing as to why we should choose Dnlib over mono.cecil, does it offer something more / better or what?

Wumbology Expert - March 31, 2014, 8:34 am

(Mar 31, 2014 08:27 AM)Dispersia Wrote: [\[CITE\]](#) I see nothing as to why we should choose Dnlib over mono.cecil, does it offer something more / better or what?

Mono.Cecil is not all perfect. It handles metadata differently than dnlib. So in some situations, Mono.Cecil may create generic detections while dnlib may not.

#RockY - March 31, 2014, 8:39 am

Thanks for the tut it really works :)

ubbelol - March 31, 2014, 9:16 am

(Mar 31, 2014 08:27 AM)Dispersia Wrote: [\[CITE\]](#) I see nothing as to why we should choose Dnlib over mono.cecil, does it offer something more / better or what?

Dnlib was made by 0xd4d (creator of de4dot) to be much more robust and stable loading invalid metadata and obfuscated files.

"0xd4d@bitbucket Wrote:dnlib is a library that can read, write and create .NET assemblies and modules.

It was written for de4dot which must have a rock solid assembly reader and writer library since it has to deal with heavily obfuscated assemblies with invalid metadata. If the CLR can load the assembly, dnlib must be able to read it and save it.

Features:

- Supports reading, writing and creating .NET assemblies/modules targeting any .NET framework (eg. desktop, Silverlight, Windows Phone, etc).
- Supports reading and writing mixed mode assemblies (eg. C++/CLI)
- Can read and write non-ECMA compatible .NET assemblies that MS' CLR can load and execute
- Very stable and can handle obfuscated assemblies that crash other similar libraries.
- High and low level access to the metadata
- Output size of non-obfuscated assemblies is usually smaller than the original assembly
- Metadata tokens and heaps can be preserved when saving an assembly
- Assembly reader has hooks for decrypting methods and strings
- Assembly writer has hooks for various writer events
- Easy to port code from Mono.Cecil to dnlib
- Add/delete Win32 resource blobs
- Saved assemblies can be strong name signed and enhanced strong name signed

Dispersia - March 31, 2014, 9:35 am

(Mar 31, 2014 09:16 AM)ubbelol Wrote: [\[CITE\]](#) Dnlib was made by 0xd4d (creator of de4dot) to be much more robust and stable loading invalid metadata and obfuscated files.
ahh, thank you for the answer and clearing that up.

Wumbology Expert - March 31, 2014, 12:32 pm

(Mar 31, 2014 09:16 AM)ubbelol Wrote: [\[CITE\]](#) Dnlib was made by 0xd4d (creator of de4dot) to be much more robust and stable loading invalid metadata and obfuscated files.

Thanks for clearing this up. I wasn't sure on the exact difference, but I know on the dnlib bitbucket it says something about handling metadata.

Mahmoudnia - March 31, 2017, 10:52 pm

I follow these instructions but my variable's name is null and I can't rename that. for example I have 2 variables in Form1_Load I can read count of variables but name is null.

This is my code

PHP Code

```
<?php
foreach (MethodDef method in type.Methods)
{
    //my commands

    if (method.HasBody && method.Body.HasVariables)
    {
        foreach (var vr in method.Body.Variables)
        {
            vr.Name = GenerateName();
        }
    }
}
```

arsium - October 24, 2020, 3:38 am

(Mar 31, 2017 10:52 PM)Mahmoudnia Wrote: [\[CITE\]](#) I follow these instructions but my variable's name is null and I can't rename that. for example I have 2 variables in Form1_Load I can read count of variables but name is null.

This is my code

PHP Code

```
<?php
foreach (MethodDef method in type.Methods)
{
    //my commands

    if (method.HasBody && method.Body.HasVariables)
    {
        foreach (var vr in method.Body.Variables)
        {
            vr.Name = GenerateName();
        }
    }
}
```

I got that too !

sh@rp - October 24, 2020, 6:21 am

Variable names are stored in pdb files ... that's why you can't retrieved them in the executable itself ;)

Keep in mind that variable are dynamically generated during execution

Modified those variables without distributing the exe and pdb file in the same time is useless.... but it's not recommended to distribute pdb files after assembly obfuscation.

++