med0x2e / vba2clr Public

med0x2e Update README.md 09df403 · 2 years ago 15 Commits

ExecuteAssembly.clr.x.vba	Added support for CLR 4.0 & .NET 4.x ...	2 years ago
ExecuteAssembly.clr2.0.vba	Additional details (CLR 2.0) support only.	2 years ago
HelloWorld.exe	Add files via upload	2 years ago
README.md	Update README.md	2 years ago
_AppDomain.idl.not.cs	Create _AppDomain.idl.not.cs	2 years ago
vba2clr.b64.vba	Create vba2clr.b64.vba	2 years ago
vba2clr.hex.vba	Create vba2clr.hex.vba	2 years ago

README

TLDR:

Just experimenting with different ways to load CLR (.NET) assemblies on VBA locally and remotely by using `AppDomain.ExecuteAssembly` or any other handy method after getting arround `AccessVBOM` (programmatic access to visual basic project is not trusted) from VBA.

- `vba2clr.*.vba` :
  - Sets `AccessVBOM` regkey to 1
  - Instantiate a `Word.Application` COM object. (could be `Excel.Application` MS PowerPoint, Access ..etc).
  - Add Macro From String (Macro corresponds to b64/hex encoded ExecuteAssembly.vba)
  - Run ExecuteAssembly.vba Macro using `wordObj.Application.Run...`

.NET from VBA:

- `ExecuteAssembly.clr.2.0.vba` : Up to .NET 3.5
  - Adds the required mscordlib references
  - Instantiates the required objects (`IDomain` , `ICRHost`)
  - Pack the required `AppDomain.ExecuteAssembly` arguments into two separate arrays (variables, types).
  - Use DispCallFunc to call `AppDomain.ExecuteAssembly(Arg1, Arg2)` (VFTable offset 51) where `Arg1` is the ".NET Assembly URL" or "Local Path" and `Arg2` is the return value.
  - AppDomain methods VFTable offsets can be checked on the AppDomain IDL _AppDomain.idl, just keep in mind that the AppDomain interface inherits from the IUnknown interface, so functions/methods VTable offsets start from the third offset onwards, this is because interfaces inheriting from IUnknown have the first 3 entries in their vtable set to `QueryInterface` , `AddRef` , `Release` methods.

About

Running .NET from VBA

Readme
Activity
131 stars
3 watching
19 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

VBA 54.2% C# 45.8%

- WinDbg or IDA can be also used as alternatives for extracting functions/methods VTable offsets.

- `ExecuteAssembly.clr.x.vba` : supports .NET 2, 3.5 and 4.x

OPSEC Notes:

- Creating a COM object for `Word.Application` (or `Excel.Application` ..etc), will result spawning an additional `WinWord.exe` as a child process of svchost.exe instead of the main `WinWord.exe` process.
- `AccessVBOM` Registry key is modified/restored via COM using `WScript.Shell` , using win32 APIs could be a better alternative.
- Hosting the CLR using win32 APIs on VBA is obviously safer than updating the `AccessVBOM` registry key, will leave this for an other day...
- Other .NET APIs such as `System.CodeDom.Compiler` can be used to compile/execute c# code from VBA, check reference below;

References:

- https://github.com/jet2jet/vb2clr
- https://github.com/med0x2e/NET-Assembly-Inject-Remote