SANS

Oh, You Silly Framework!: An Intro to Analyzing .NET Malware

SANS Sydney 2020

Ryan Chapman

BlackBerry® Intelligent Security. Everywhere.

SANS @Mic Webcast

Ryan Chapman (@rj_chap)

Principal Incident Response Analyst @ BlackBerry

- Full-time trainer → SOC → CIRT → Consulting
- DFIR / Malware Analysis (Blue Team!)
- SANS Instructor
  - FOR610: Reverse Engineering Malware
- Lead Organizer for CactusCon
- PluralSight Author
- **incidentresponse.training**

Agenda

We only have 1hr – Let's make the best of it!

- .NET Overview
- Code Compilation Overview
- Tools of the Trade
- Initial Static Analysis
- **Demo**: Simple .NET Program Analysis
- Obfuscation in .NET
- **Demo**: AgentTesla Stage 1 Decoding
- Questions / Wrap-Up

.NET Overview

Pronounced "Dot Net"

- A Microsoft platform for multi-OS/device development, consisting of:
- **.NET Framework**
  - Collection of technologies to build and run Windows apps
- .NET Core
  - Cross-platform implementation of .NET
- Mono/Xamarin
  - .NET implementation for various systems, including Android & iOS
  - e.g. Using NetworkMiner in Linux
- Interested in Learning .NET?

.NET Framework Overview

Implements a multitude of languages and [un]managed code

- .NET programming languages
  - C# (“C sharp”)
  - F# (“F sharp”)
  - Visual Basic
  - J# (“J sharp”) support decom’d in 2017

- Other languages via “unmanaged code”
  - e.g. An unmanaged C++ runtime

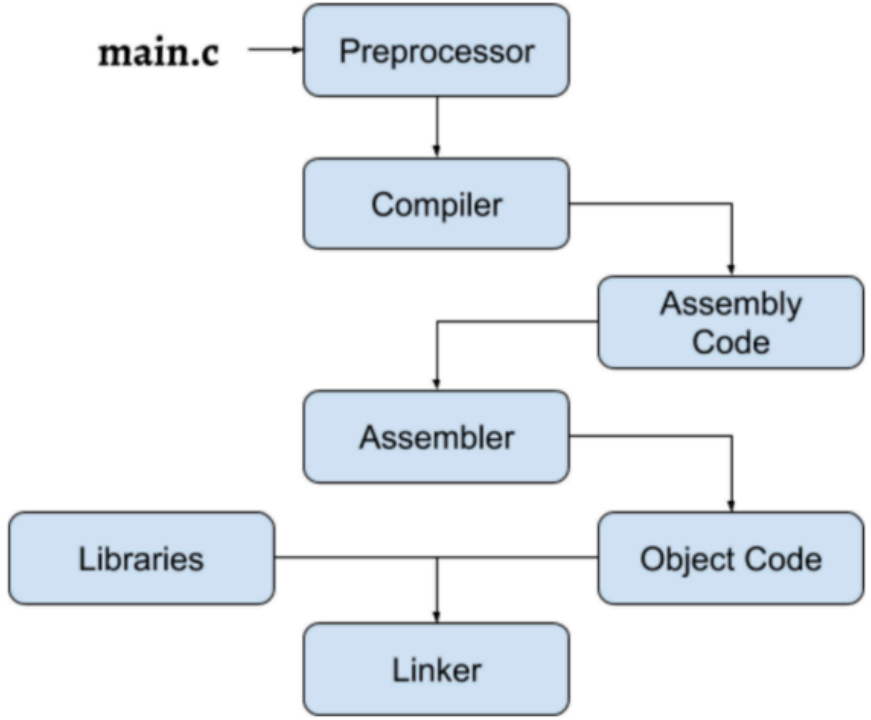
.NET Framework Components

Managed code basics

- "Managed code" runs within the **Common Language Runtime (CLR)**
  - Implements garbage collection, type checking, exception handling, and more
- Relies on the Common Intermediate Language (CIL)
  - Binary instructions defined in the Common Language Infrastructure (CLI) specs
  - a.k.a. Microsoft Intermediate Language (MSIL), or just IL
- .NET Libraries
  - Framework Class Library (FCL) organized into "namespaces" (e.g. `System.*`)
  - FCL implements Base Class Library (BCL)
  - BCL classes mostly live in `mscorlib.dll`, `System.dll`, and `System.Core.dll`

General Code Compilation

General C compiler example

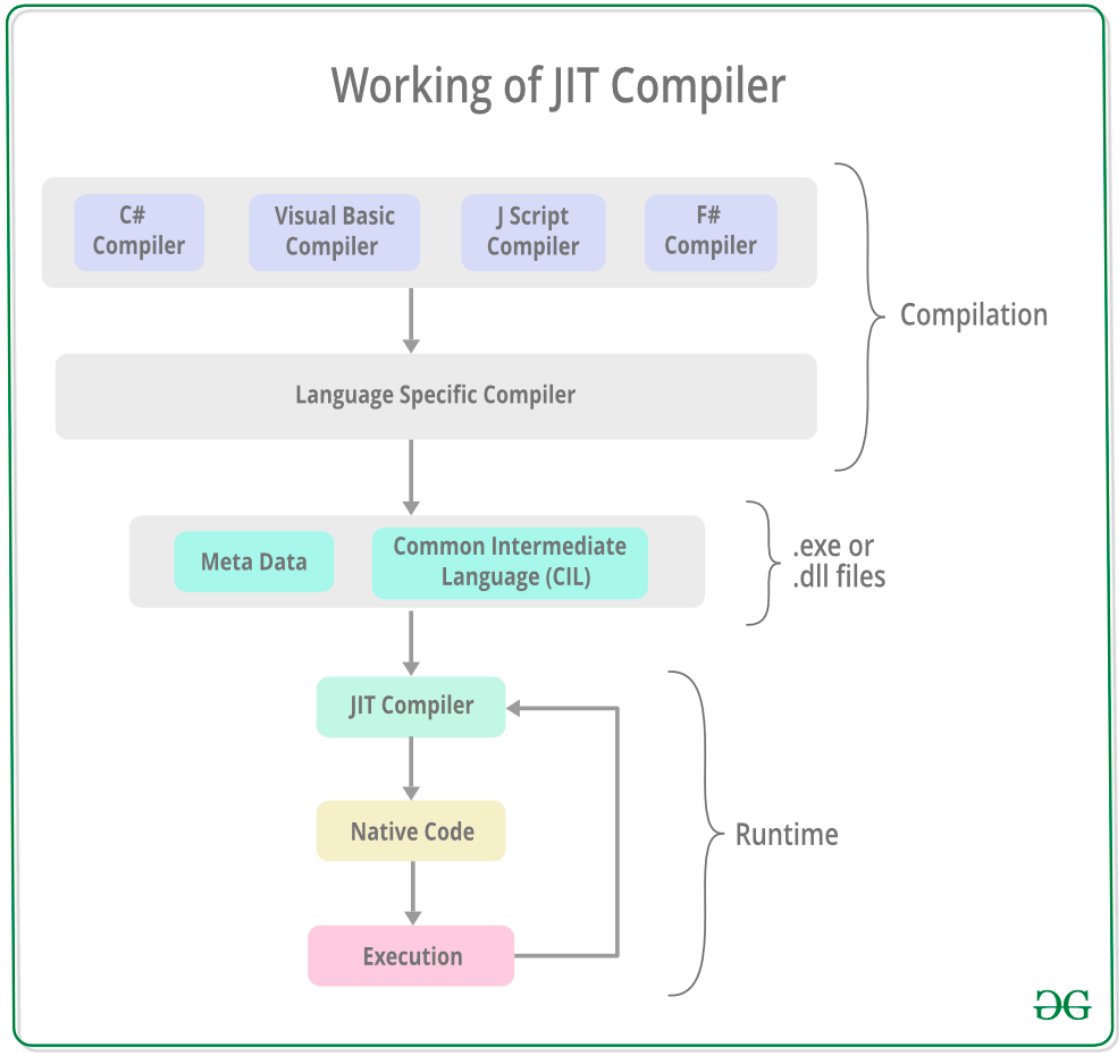


Anthony Huggins, 2019

.NET Code Compilation

Just-in-time (JIT) compilation

- Initial compiler creates CIL code
- Compiled CIL stored in PE format
- On execution, CLR takes over
- CIL turned into machine code
- CLR → Virtual Execution System (VES)
- Managed code executed on CPU

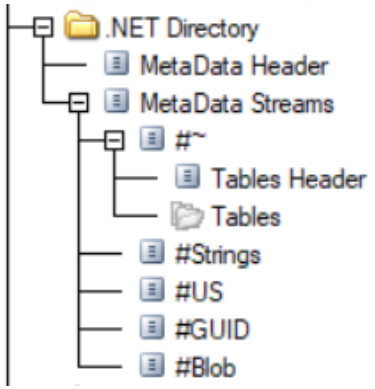


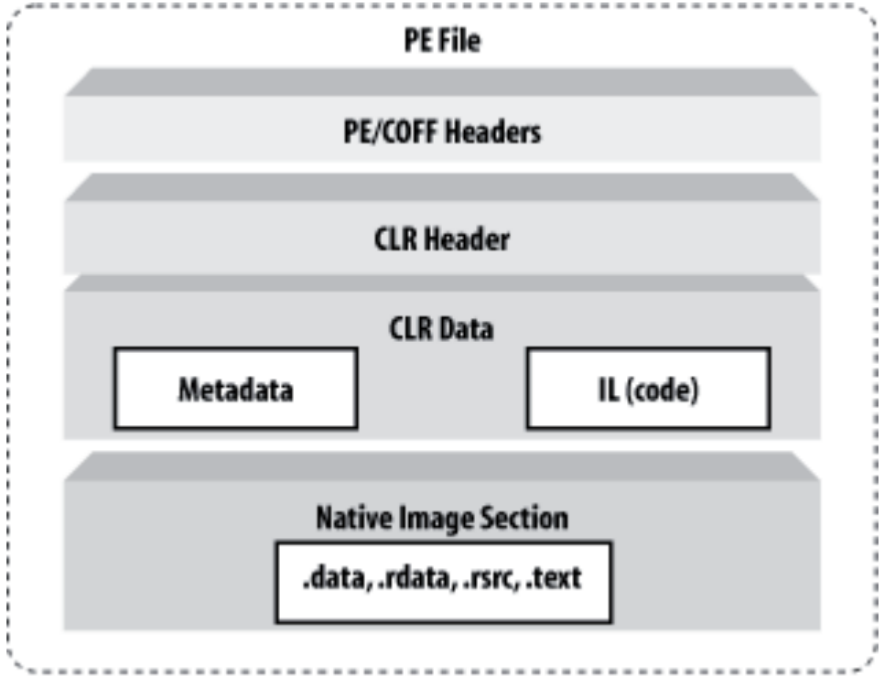
GeeksforGeeks, n.d.

.NET Code Compilation cont.

.NET PE files include CLR Header & CLR Data

- .NET File Format explained
- CLR-specific headers
- Can also contain unmanaged code
- CLR Metadata useful for analysis





Pontiroli & Martinez, 2015

Tools of the Trade

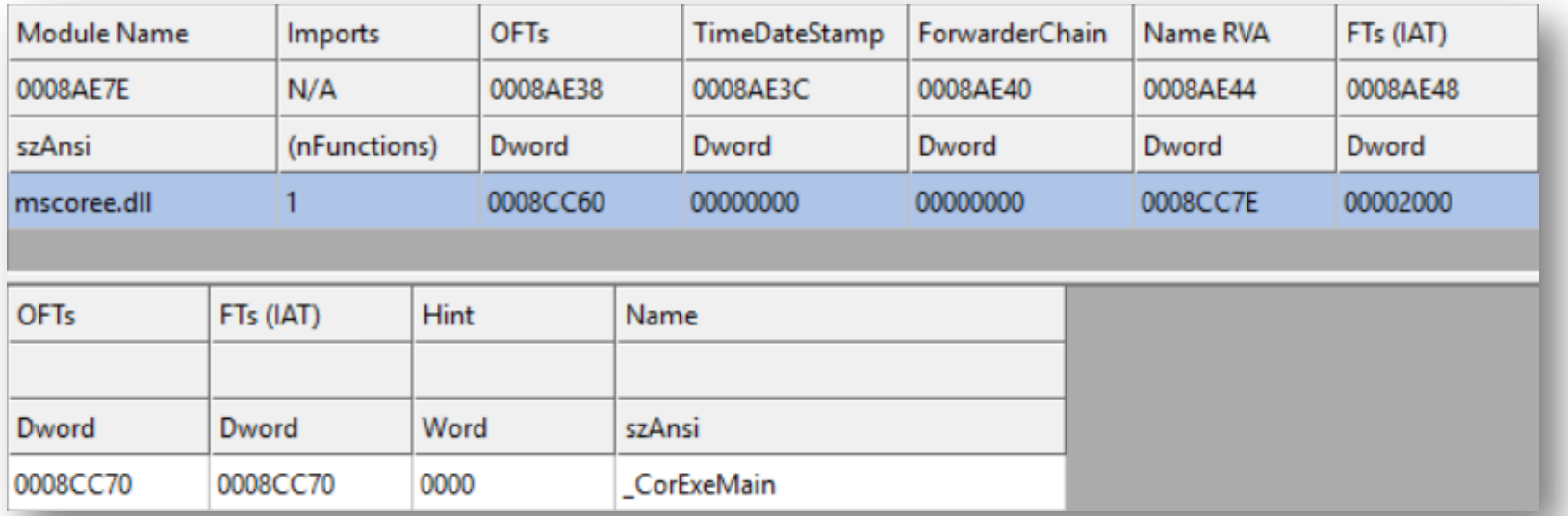
Popular .NET analysis tools

- General PE tools
  - **CFF Explorer**, pestudio, Exeinfo, Detect-It-Easy
- ILSpy
  - FOSS .NET decompiler
- **dnSpy**
  - FOSS debugger and editor (fork of ILSpy)
- **de4dot**
  - FOSS .NET deobfuscator and unpacker
- Others
  - DotPeek, Reflexil, .NET Reflector, .NET Fiddle, Visual Studio, IDA/Ghidra, and more

Initial Static Analysis

Ways to determine if a PE is .NET

- Imports
  - mscoree.dll
    - _CorExeMain
    - _CorDllMain
- Directories
- Strings
- Manifest



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0008AE7E	N/A	0008AE38	0008AE3C	0008AE40	0008AE44	0008AE48
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
mscoree.dll	1	0008CC60	00000000	00000000	0008CC7E	00002000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
0008CC70	0008CC70	0000	_CorExeMain

The above will be reviewed using our first sample, up next!

Simple .NET Program Analysis

Shall we play a game?

- FireEye's 2019 Flare-On Challenge Solutions
  - Samples linked above
- Flare-On 6: Challenge 1 – MemecatBattlestation.exe
  - Written by Nick Harbour
- Great example of code analysis requirements
- VT: 22c3ad804d02c689203426d020f3cf86a0190f40

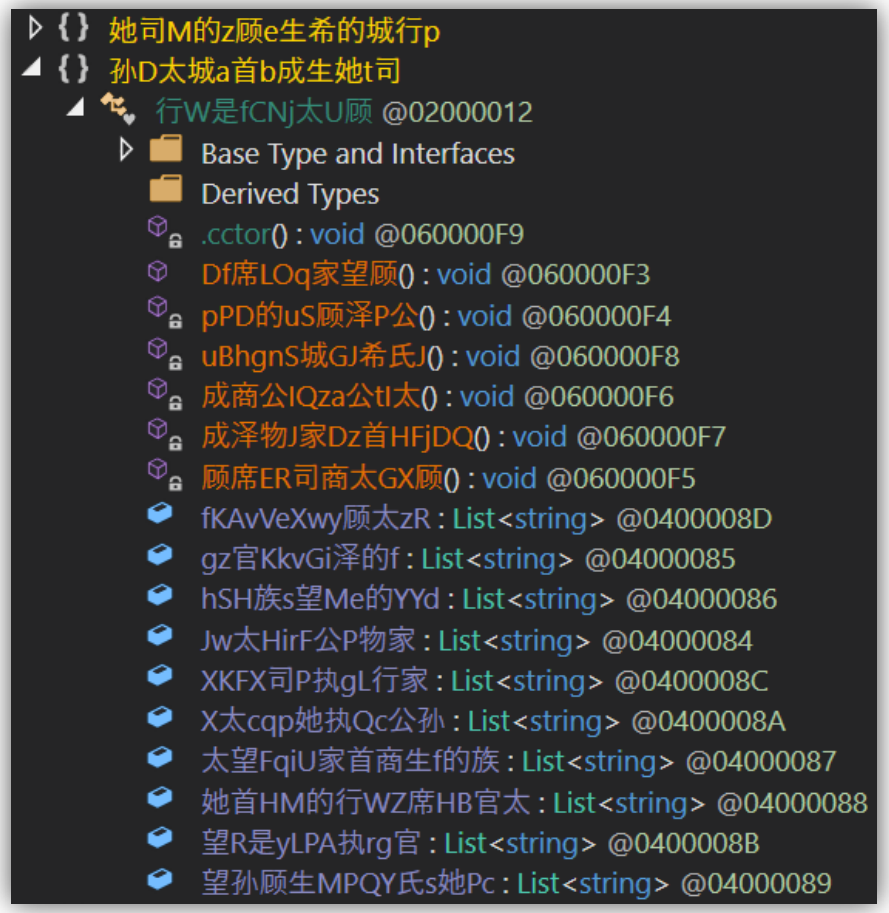
Nick Harbour, 2019

DEMO

Obfuscation in .NET

If decompilation is so easy… what's the rub?

- .NET Obfuscation frameworks
  - Commercial, free, FOSS, & custom
- Example obfuscators
  - .NET Reactor – Commercial
  - Dotfuscator – Commercial
  - ConfuserEx – FOSS
- Be on the lookout for:
  - System.Reflection.Assembly.**Load()**
  - System.Reflection.Assembly.**LoadFile()**
  - System.Reflection.MethodInfo.**Invoke()**



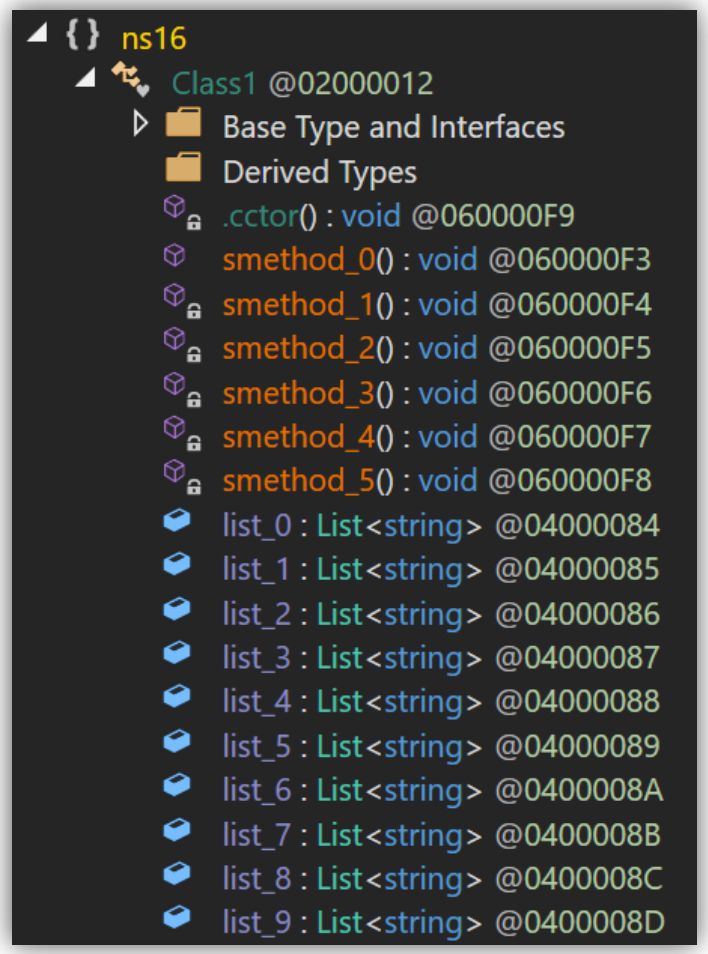
Obfuscated AgentTesla sample

Dealing with Obfuscation in .NET

The easy button + focus points

- de4dot is a fantastic deobfuscator
  - Supports ~20 known obfuscators/packers
  - **Can rename variables and symbols**
    - Even if obfuscator is unknown

- For initial stages, hunt byte arrays!
  - Byte arrays are used *often* to deobfuscate data
  - Look for `byte[]`
  - Example coming up!



Deobfuscated AgentTesla sample

AgentTesla Stage 1 Analysis

AgentTesla sample from ~2 months ago

- Popular .NET-based Remote Access Trojan (RAT)
  - Been around since at least 2014
- Newer samples have been using a Stage 1 dropper
- VT: c953573a155ee13aa186147e5d771e0a6b9f41cc

DEMO

SANS

Thank You!!
Questions? Comments?

@rj_chap
incidentresponse.training