Threat hunting .NET malware with YARA

Santiago Martin Pontiroli

Global Research and Analysis Team

Kaspersky Lab

@spontiroli

#BSIDESNYC

GREAT

KASPERSKY lab

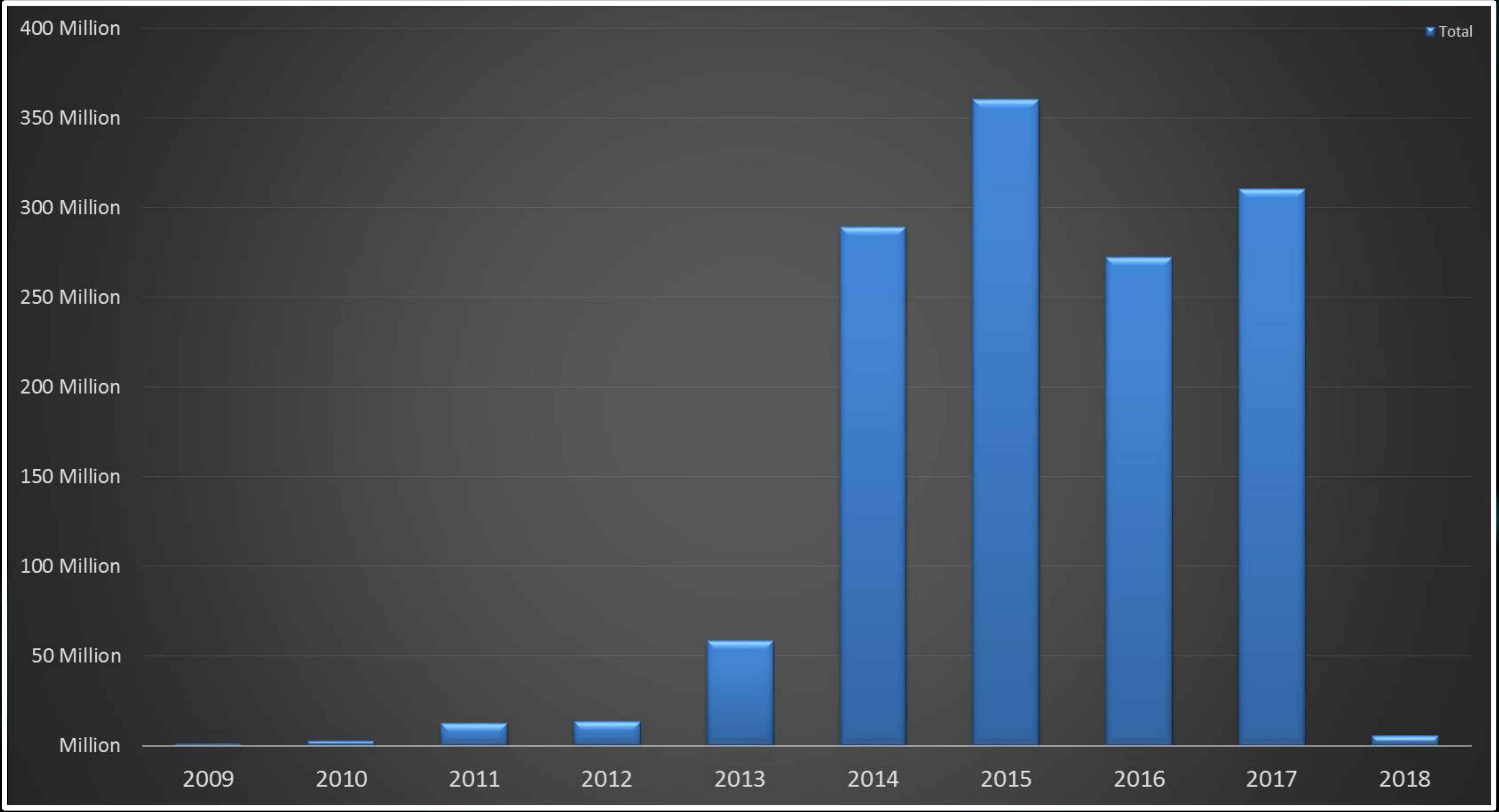
What's the problem with .NET malware?

- Available by default in most Windows installations, it has become the de-facto standard for software development in Microsoft's family of operating systems.

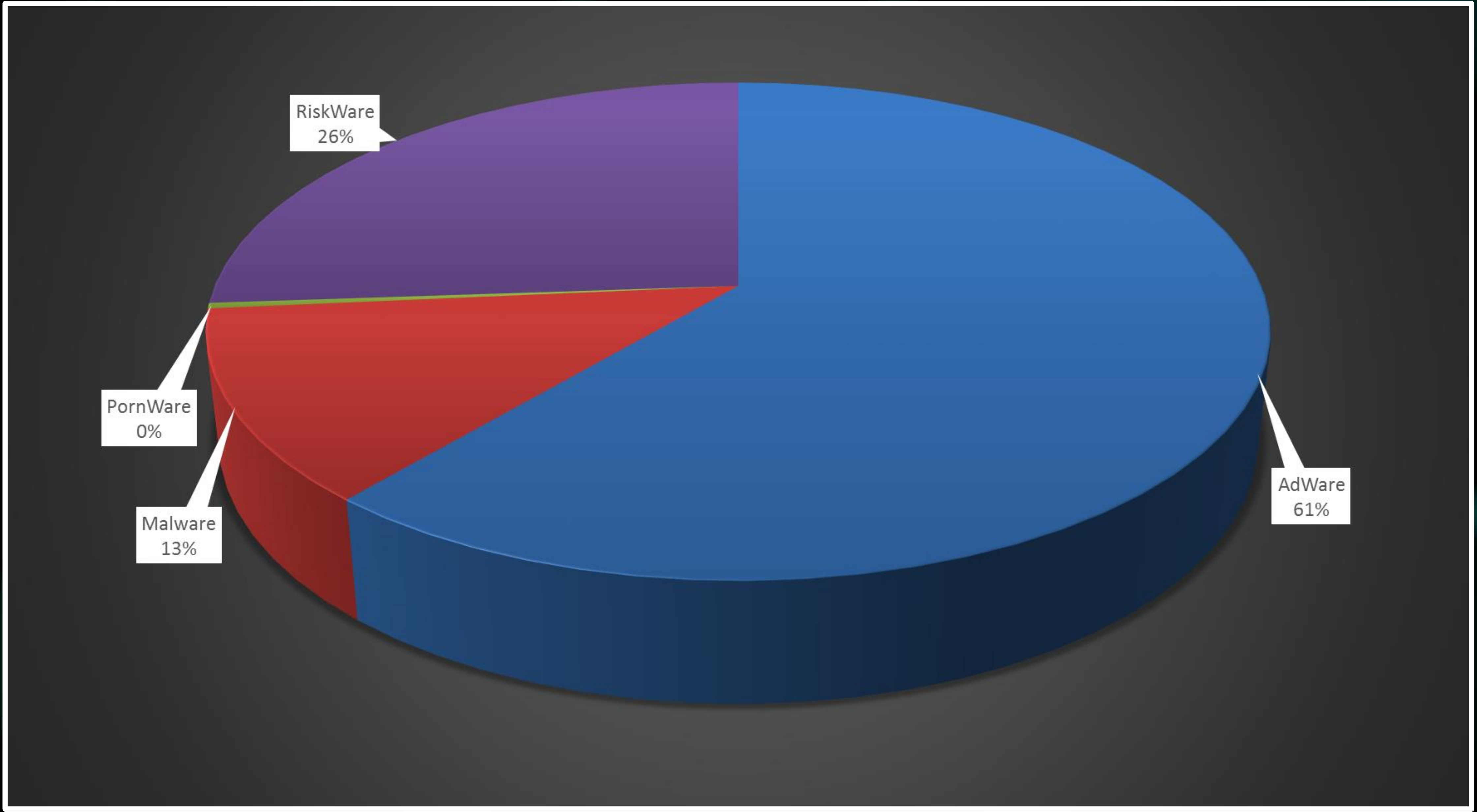
- Vast amounts of ready-to-use functionality make .NET and PowerShell a deadly combination at the hands of cybercriminals.

- Since 2009 there has been a steady growth in the number of .NET malware, but it's still treated as other regular PEs by analysts.

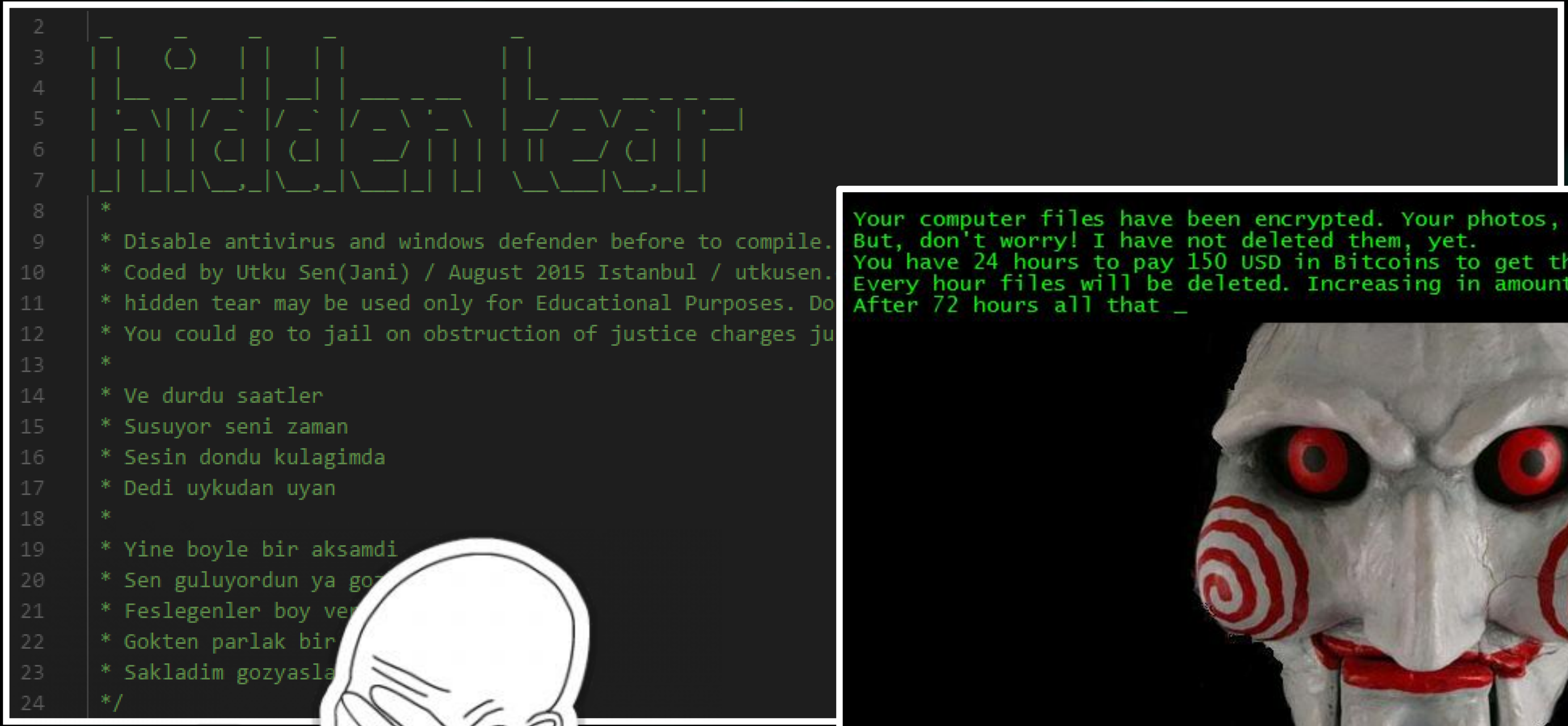
The .NET malware ecosystem as it is today, yearly detections



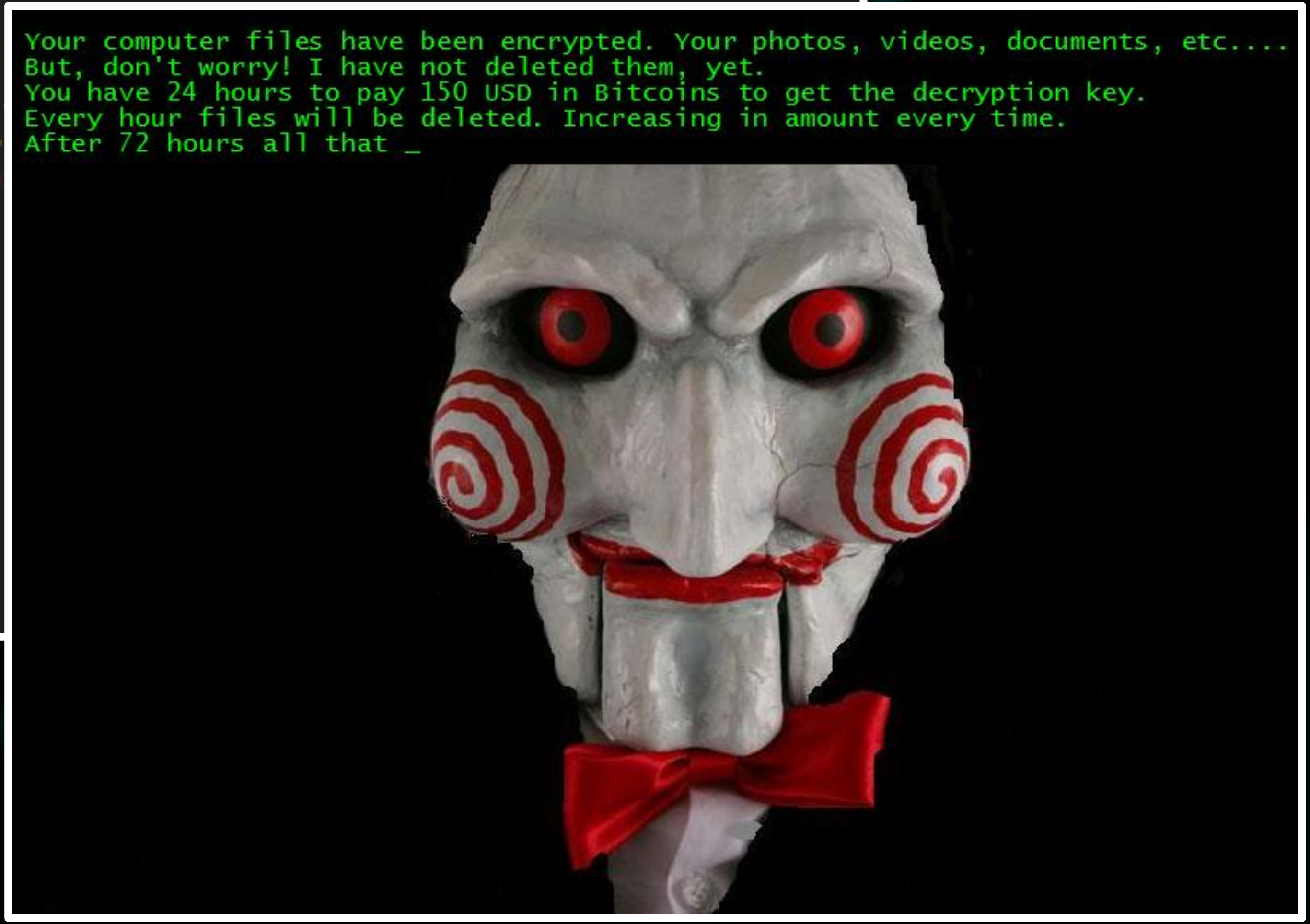
The .NET malware ecosystem as it is today, detections by type



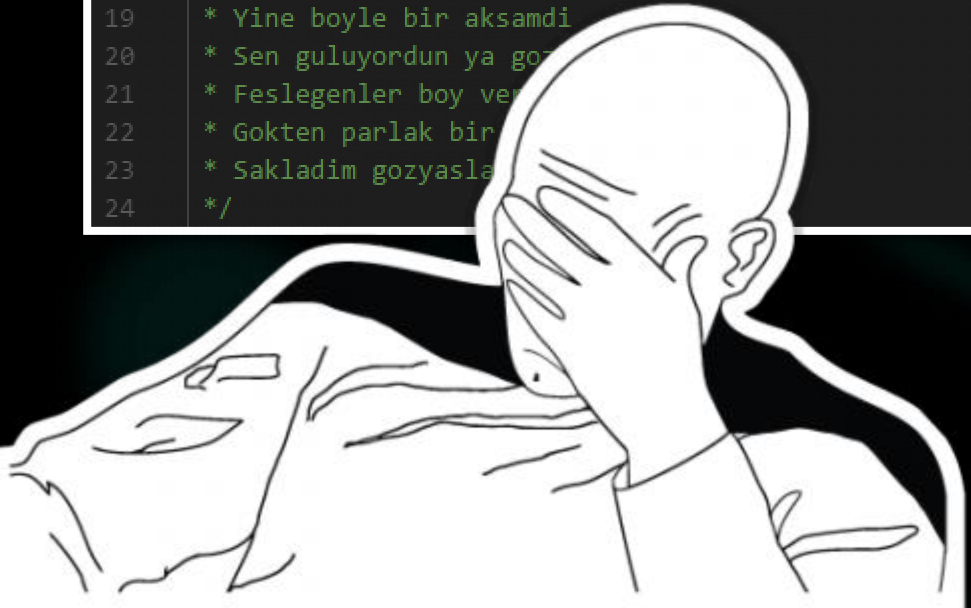
From the most simple to targeted attacks, .NET malware is rising



```
 _     _     _     _              _
| |   (_)   | |   | |            | |
| |__  _  __| | __| | ___ _ __   | |_ ___  __ _ _ __
| '_ \| |/ _` |/ _` |/ _ \ '_ \  | __/ _ \/ _` | '__|
| | | | | (_| | (_| |  __/ | | | | ||  __/ (_| | |
|_| |_|_|\__,_|\__,_|\___|_| |_|  \__\___|\__,_|_|
 *
 * Disable antivirus and windows defender before to compile.
 * Coded by Utku Sen(Jani) / August 2015 Istanbul / utkusen.
 * hidden tear may be used only for Educational Purposes. Do
 * You could go to jail on obstruction of justice charges ju
 *
 * Ve durdu saatler
 * Susuyor seni zaman
 * Sesin dondu kulagimda
 * Dedi uykudan uyan
 *
 * Yine boyle bir aksamdi
 * Sen guluyordun ya goz
 * Feslegenler boy ver
 * Gokten parlak bir
 * Sakladim gozyasla
 */
```

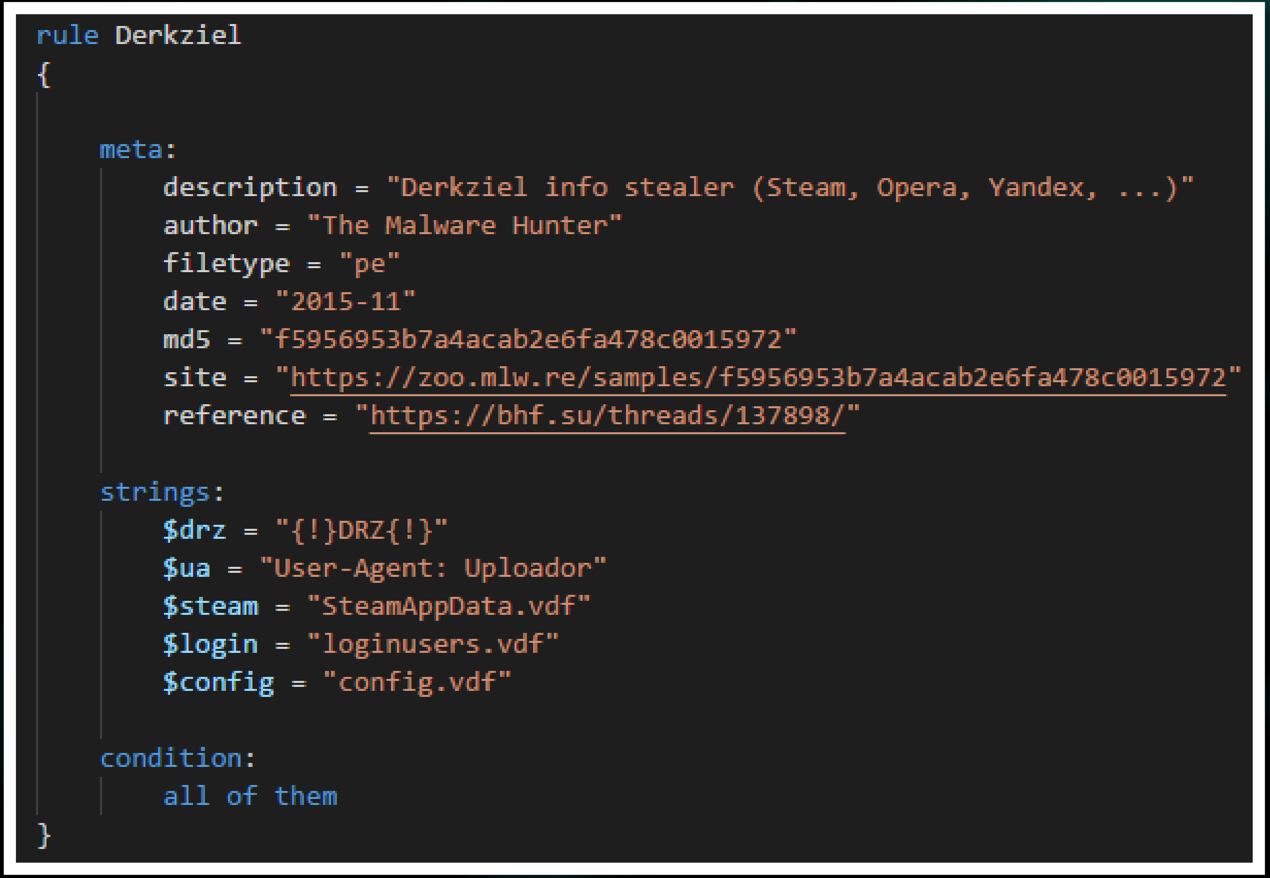


```
Your computer files have been encrypted. Your photos, videos, documents, etc....
But, don't worry! I have not deleted them, yet.
You have 24 hours to pay 150 USD in Bitcoins to get the decryption key.
Every hour files will be deleted. Increasing in amount every time.
After 72 hours all that _
```



YARA, the pattern matching swiss knife

YARA helps malware researchers **identify and classify** malware samples by using descriptive patterns.

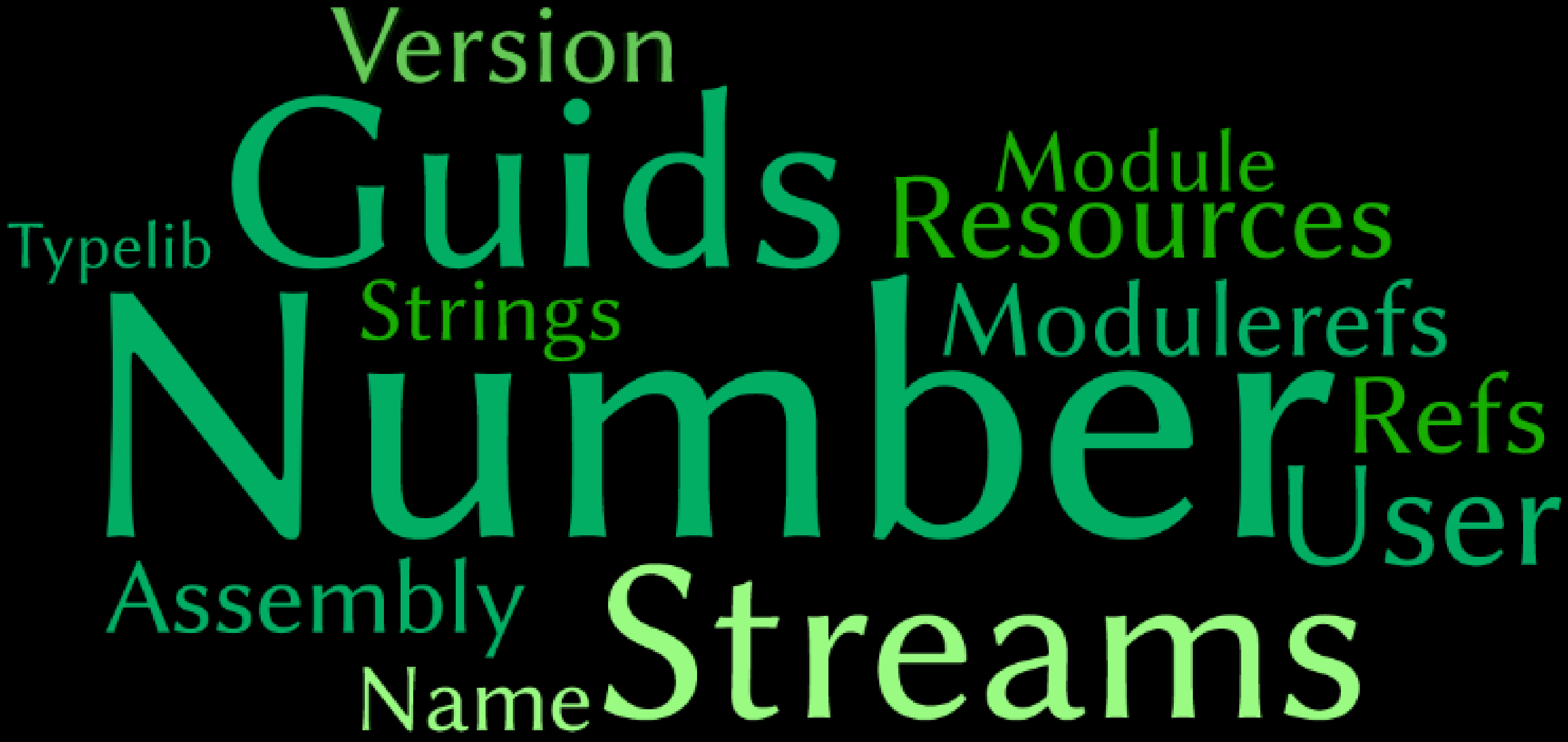


```
rule Derkziel
{

    meta:
        description = "Derkziel info stealer (Steam, Opera, Yandex, ...)"
        author = "The Malware Hunter"
        filetype = "pe"
        date = "2015-11"
        md5 = "f5956953b7a4acab2e6fa478c0015972"
        site = "https://zoo.mlw.re/samples/f5956953b7a4acab2e6fa478c0015972"
        reference = "https://bhf.su/threads/137898/"

    strings:
        $drz = "{!}DRZ{!}"
        $ua = "User-Agent: Uploador"
        $steam = "SteamAppData.vdf"
        $login = "loginusers.vdf"
        $config = "config.vdf"

    condition:
        all of them
}
```

- You need a **set of strings** or a **logical condition** to build your rule.
- YARA contains several **modules for extending its features** and expressing more complex conditions.

YARA and .NET, getting better by the day

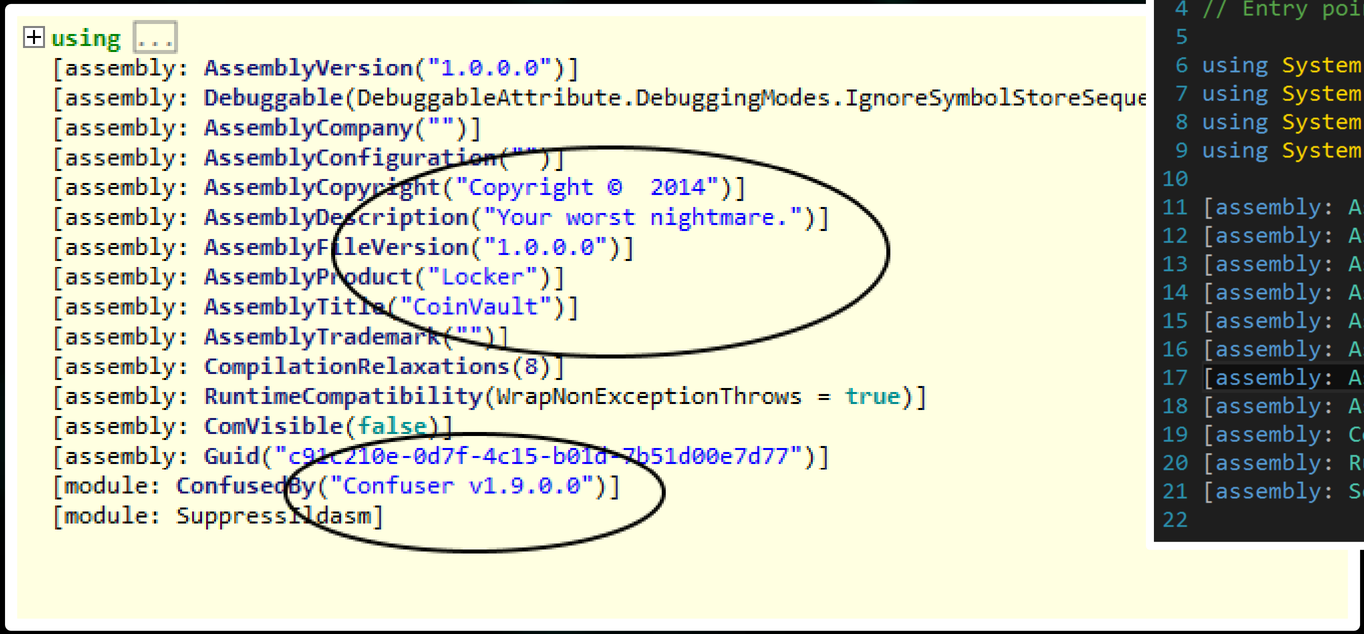
The dotnet module allows you to create more fine-grained rules for .NET files by using attributes and features of the .NET file format.



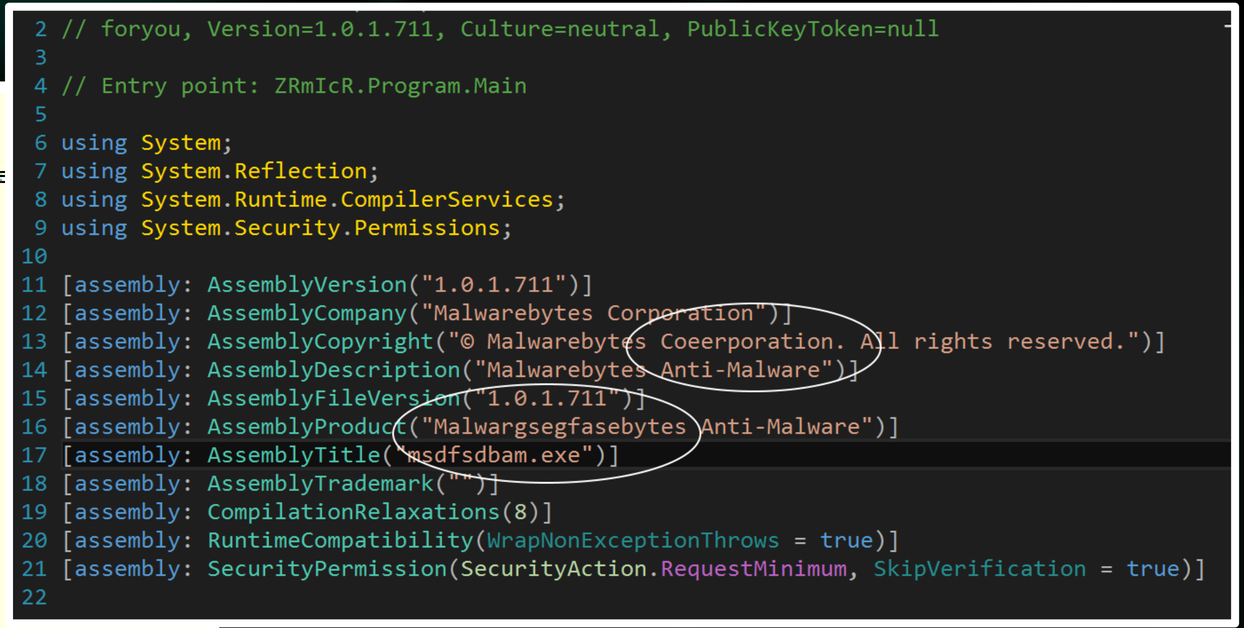
Why think about .NET malware differently?

In general, MSIL binaries are easier to reverse engineer than compiled code, however obfuscation makes some executables useless by reducing the amount of indicators we can gather from them statically.

Using additional information embedded into the binary then becomes useful for finding related samples in Virus Total and similar engines.

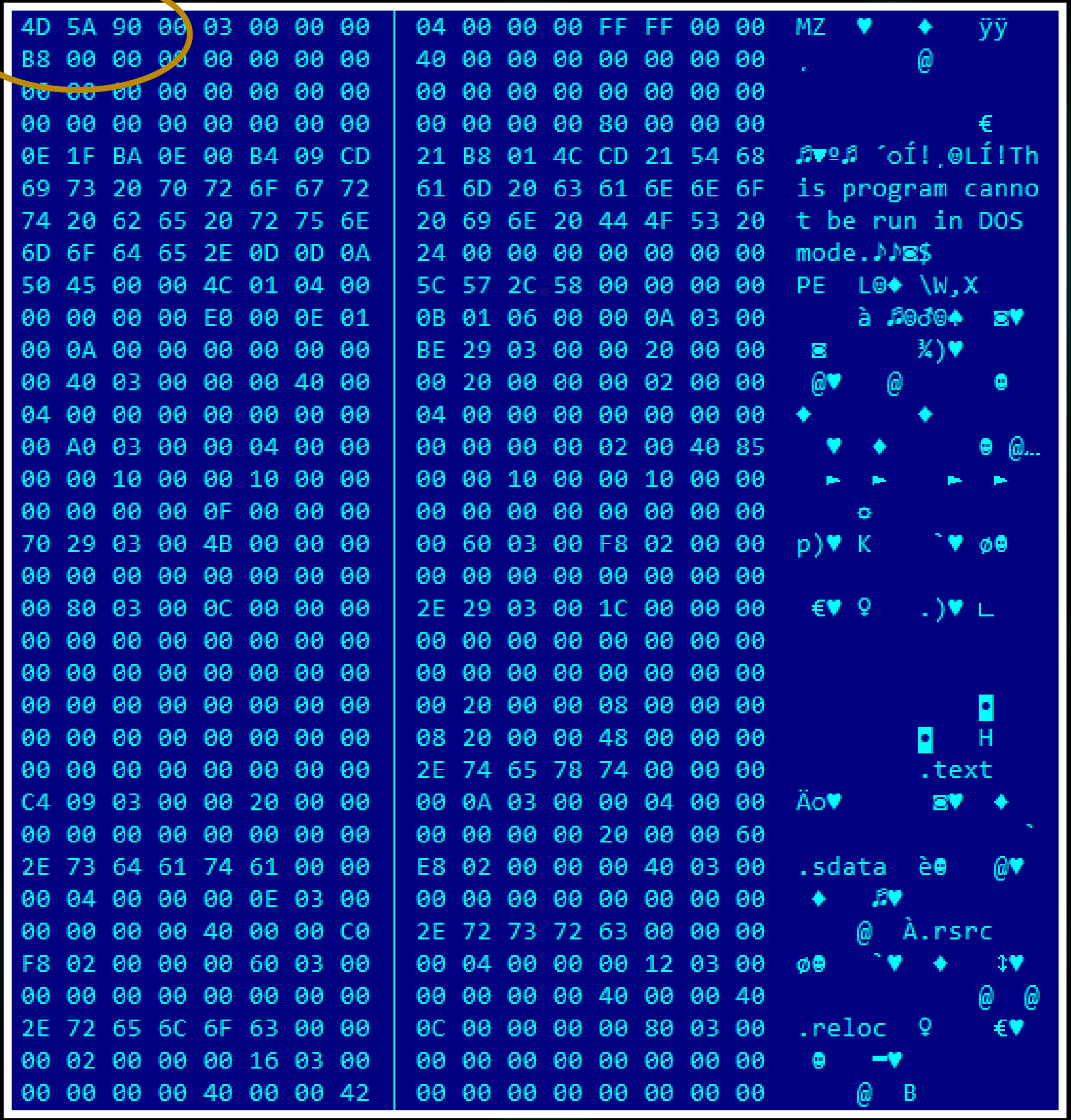


```
+ using ...
  [assembly: AssemblyVersion("1.0.0.0")]
  [assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSeque
  [assembly: AssemblyCompany("")]
  [assembly: AssemblyConfiguration("")]
  [assembly: AssemblyCopyright("Copyright ©  2014")]
  [assembly: AssemblyDescription("Your worst nightmare.")]
  [assembly: AssemblyFileVersion("1.0.0.0")]
  [assembly: AssemblyProduct("Locker")]
  [assembly: AssemblyTitle("CoinVault")]
  [assembly: AssemblyTrademark("")]
  [assembly: CompilationRelaxations(8)]
  [assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
  [assembly: ComVisible(false)]
  [assembly: Guid("c91c210e-0d7f-4c15-b01d-7b51d00e7d77")]
  [module: ConfusedBy("Confuser v1.9.0.0")]
  [module: SuppressIldasm]
```

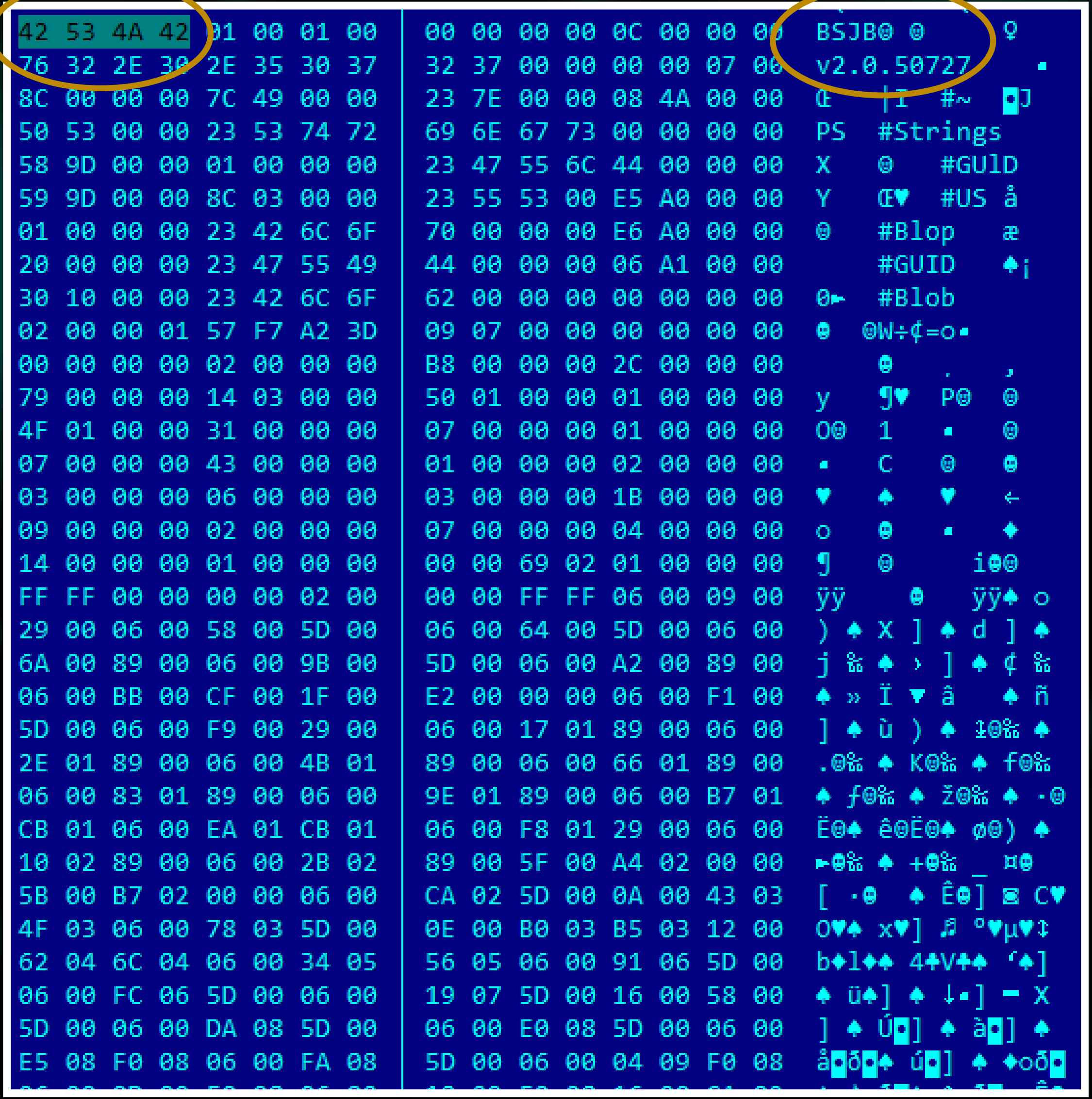


```
// foryou, Version=1.0.1.711, Culture=neutral, PublicKeyToken=null

// Entry point: ZRmIcR.Program.Main

using System;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Security.Permissions;

[assembly: AssemblyVersion("1.0.1.711")]
[assembly: AssemblyCompany("Malwarebytes Corporation")]
[assembly: AssemblyCopyright("© Malwarebytes Coeerporation. All rights reserved.")]
[assembly: AssemblyDescription("Malwarebytes Anti-Malware")]
[assembly: AssemblyFileVersion("1.0.1.711")]
[assembly: AssemblyProduct("Malwargsegfasebytes Anti-Malware")]
[assembly: AssemblyTitle("msdfsdbam.exe")]
[assembly: AssemblyTrademark("")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: SecurityPermission(SecurityAction.RequestMinimum, SkipVerification = true)]
```

Anatomy of a .NET Assembly, from MZ to BSJB

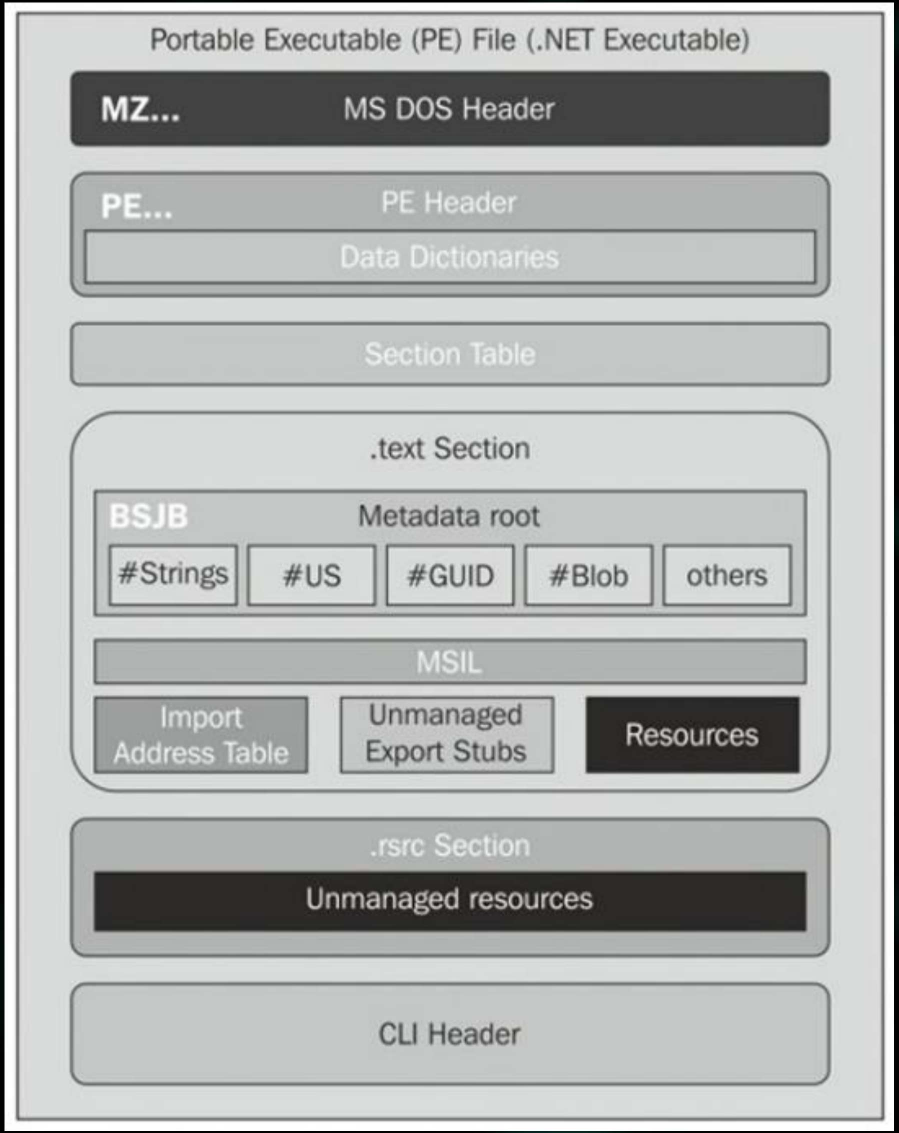


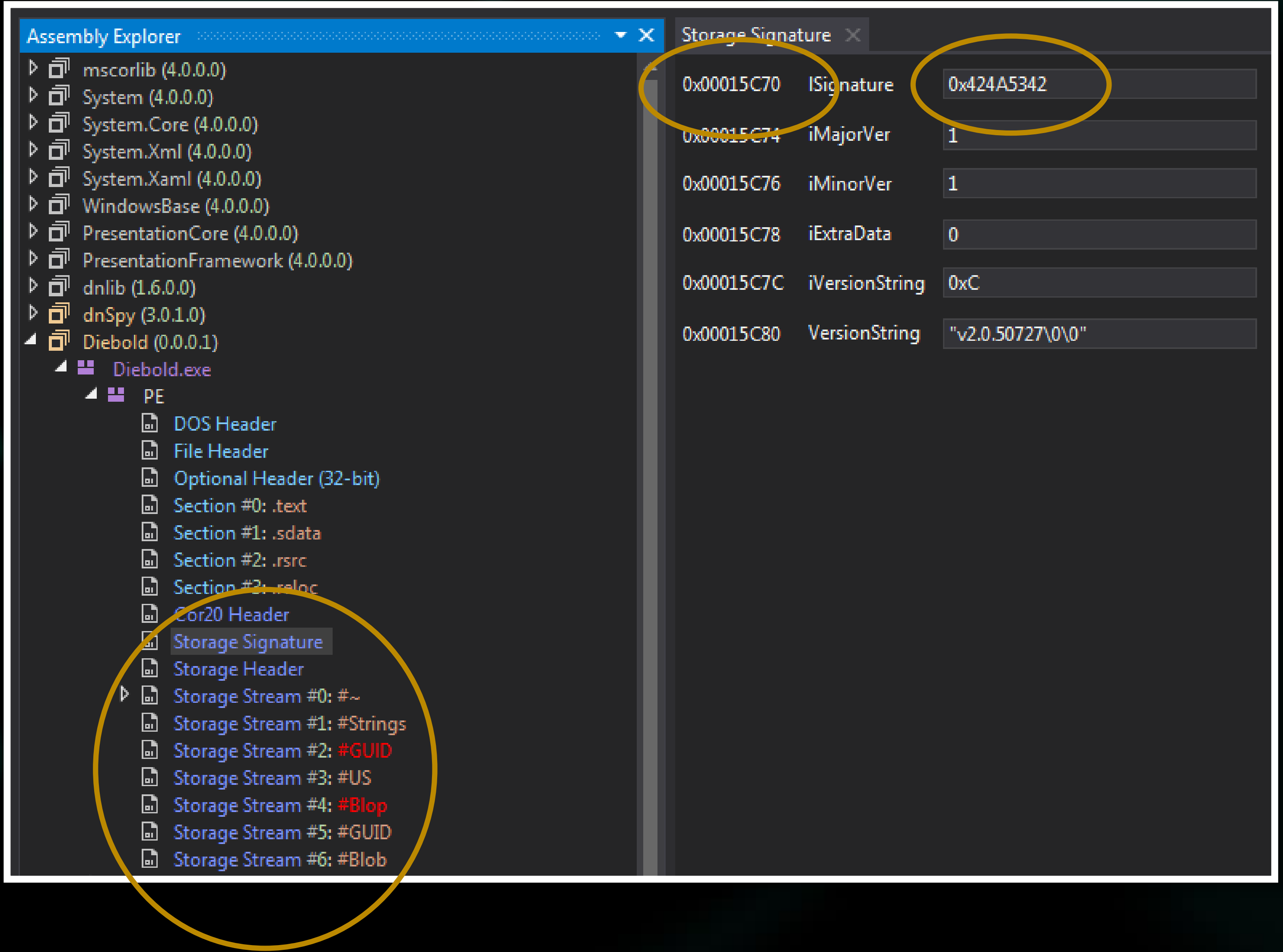
```
4D 5A 90 00 03 00 00 00   04 00 00 00 FF FF 00 00  MZ  ♥   ♦   ÿÿ
B8 00 00 00 00 00 00 00   40 00 00 00 00 00 00 00  ¸       @
00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00   00 00 00 00 80 00 00 00              €
0E 1F BA 0E 00 B4 09 CD   21 B8 01 4C CD 21 54 68  ♫▼º♫ ´oÍ!¸☺LÍ!Th
69 73 20 70 72 6F 67 72   61 6D 20 63 61 6E 6E 6F  is program canno
74 20 62 65 20 72 75 6E   20 69 6E 20 44 4F 53 20  t be run in DOS
6D 6F 64 65 2E 0D 0D 0A   24 00 00 00 00 00 00 00  mode.♪♪◙$
50 45 00 00 4C 01 04 00   5C 57 2C 58 00 00 00 00  PE  L☺♦ \W,X
00 00 00 00 E0 00 0E 01   0B 01 06 00 00 0A 03 00      à ♫☺♂☺♠  ◙♥
00 0A 00 00 00 00 00 00   BE 29 03 00 00 20 00 00   ◙       ¾)♥
00 40 03 00 00 00 40 00   00 20 00 00 00 02 00 00   @♥   @       ☻
04 00 00 00 00 00 00 00   04 00 00 00 00 00 00 00  ♦       ♦
00 A0 03 00 00 04 00 00   00 00 00 00 02 00 40 85    ♥  ♦       ☻ @…
00 00 10 00 00 10 00 00   00 00 10 00 00 10 00 00    ►  ►    ►  ►
00 00 00 00 0F 00 00 00   00 00 00 00 00 00 00 00      ☼
70 29 03 00 4B 00 00 00   00 60 03 00 F8 02 00 00  p)♥ K    `♥ ø☻
00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00 80 03 00 0C 00 00 00   2E 29 03 00 1C 00 00 00   €♥ ♀    .)♥ ∟
00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00   00 20 00 00 08 00 00 00               ◘
00 00 00 00 00 00 00 00   08 20 00 00 48 00 00 00          ◘   H
00 00 00 00 00 00 00 00   2E 74 65 78 74 00 00 00          .text
C4 09 03 00 00 20 00 00   00 0A 03 00 00 04 00 00  Äo♥       ◙♥  ♦
00 00 00 00 00 00 00 00   00 00 00 00 20 00 00 60                 `
2E 73 64 61 74 61 00 00   E8 02 00 00 00 40 03 00  .sdata  è☻   @♥
00 04 00 00 00 0E 03 00   00 00 00 00 00 00 00 00   ♦   ♫♥
00 00 00 00 40 00 00 C0   2E 72 73 72 63 00 00 00      @  À.rsrc
F8 02 00 00 00 60 03 00   00 04 00 00 00 12 03 00  ø☻   `♥  ♦   ↕♥
00 00 00 00 00 00 00 00   00 00 00 00 40 00 00 40              @  @
2E 72 65 6C 6F 63 00 00   0C 00 00 00 00 80 03 00  .reloc  ♀     €♥
00 02 00 00 00 16 03 00   00 00 00 00 00 00 00 00   ☻   ▬♥
00 00 00 00 40 00 00 42   00 00 00 00 00 00 00 00      @  B
```



```
42 53 4A 42 01 00 01 00   00 00 00 00 0C 00 00 00  BSJB☺ ☺     ♀
76 32 2E 30 2E 35 30 37   32 37 00 00 00 00 07 00  v2.0.50727    •
8C 00 00 00 7C 49 00 00   23 7E 00 00 08 4A 00 00  Œ   |I  #~  ◘J
50 53 00 00 23 53 74 72   69 6E 67 73 00 00 00 00  PS  #Strings
58 9D 00 00 01 00 00 00   23 47 55 6C 44 00 00 00  X   ☺   #GUlD
59 9D 00 00 8C 03 00 00   23 55 53 00 E5 A0 00 00  Y   Œ♥  #US å
01 00 00 00 23 42 6C 6F   70 00 00 00 E6 A0 00 00  ☺   #Blop   æ
20 00 00 00 23 47 55 49   44 00 00 00 06 A1 00 00      #GUID   ♠¡
30 10 00 00 23 42 6C 6F   62 00 00 00 00 00 00 00  0►  #Blob
02 00 00 01 57 F7 A2 3D   09 07 00 00 00 00 00 00  ☻  ☺W÷¢=○•
00 00 00 00 02 00 00 00   B8 00 00 00 2C 00 00 00      ☻   ¸   ,
79 00 00 00 14 03 00 00   50 01 00 00 01 00 00 00  y   ¶♥  P☺  ☺
4F 01 00 00 31 00 00 00   07 00 00 00 01 00 00 00  O☺  1   •   ☺
07 00 00 00 43 00 00 00   01 00 00 00 02 00 00 00  •   C   ☺   ☻
03 00 00 00 06 00 00 00   03 00 00 00 1B 00 00 00  ♥   ♠   ♥   ←
09 00 00 00 02 00 00 00   07 00 00 00 04 00 00 00  ○   ☻   •   ♦
14 00 00 00 01 00 00 00   00 00 69 02 01 00 00 00  ¶   ☺     i☻☺
FF FF 00 00 00 00 02 00   00 00 FF FF 06 00 09 00  ÿÿ    ☻   ÿÿ♠ ○
29 00 06 00 58 00 5D 00   06 00 64 00 5D 00 06 00  ) ♠ X ] ♠ d ] ♠
6A 00 89 00 06 00 9B 00   5D 00 06 00 A2 00 89 00  j ‰ ♠ › ] ♠ ¢ ‰
06 00 BB 00 CF 00 1F 00   E2 00 00 00 06 00 F1 00  ♠ » Ï ▼ â   ♠ ñ
5D 00 06 00 F9 00 29 00   06 00 17 01 89 00 06 00  ] ♠ ù ) ♠ ↨☺‰ ♠
2E 01 89 00 06 00 4B 01   89 00 06 00 66 01 89 00  .☺‰ ♠ K☺‰ ♠ f☺‰
06 00 83 01 89 00 06 00   9E 01 89 00 06 00 B7 01  ♠ ƒ☺‰ ♠ ž☺‰ ♠ ·☺
CB 01 06 00 EA 01 CB 01   06 00 F8 01 29 00 06 00  Ë☺♠ ê☺Ë☺♠ ø☺) ♠
10 02 89 00 06 00 2B 02   89 00 5F 00 A4 02 00 00  ►☻‰ ♠ +☻‰ _ ¤☻
5B 00 B7 02 00 00 06 00   CA 02 5D 00 0A 00 43 03  [ ·☻  ♠ Ê☻] ◙ C♥
4F 03 06 00 78 03 5D 00   0E 00 B0 03 B5 03 12 00  O♥♠ x♥] ♫ °♥µ♥↕
62 04 6C 04 06 00 34 05   56 05 06 00 91 06 5D 00  b♦l♦♠ 4♣V♣♠ '♠]
06 00 FC 06 5D 00 06 00   19 07 5D 00 16 00 58 00  ♠ ü♠] ♠ ↓•] ▬ X
5D 00 06 00 DA 08 5D 00   06 00 E0 08 5D 00 06 00  ] ♠ Ú◘] ♠ à◘] ♠
E5 08 F0 08 06 00 FA 08   5D 00 06 00 04 09 F0 08  å◘ð◘♠ ú◘] ♠ ♦○ð◘
```

Anatomy of a .NET Assembly, plenty of metadata to use




Assembly Explorer

- mscorlib (4.0.0.0)
- System (4.0.0.0)
- System.Core (4.0.0.0)
- System.Xml (4.0.0.0)
- System.Xaml (4.0.0.0)
- WindowsBase (4.0.0.0)
- PresentationCore (4.0.0.0)
- PresentationFramework (4.0.0.0)
- dnlib (1.6.0.0)
- dnSpy (3.0.1.0)
- Diebold (0.0.0.1)
  - Diebold.exe
    - PE
      - DOS Header
      - File Header
      - Optional Header (32-bit)
      - Section #0: .text
      - Section #1: .sdata
      - Section #2: .rsrc
      - Section #3: .reloc
      - Cor20 Header
      - Storage Signature
      - Storage Header
      - Storage Stream #0: #~
      - Storage Stream #1: #Strings
      - Storage Stream #2: #GUID
      - Storage Stream #3: #US
      - Storage Stream #4: #Blop
      - Storage Stream #5: #GUID
      - Storage Stream #6: #Blob

Storage Signature

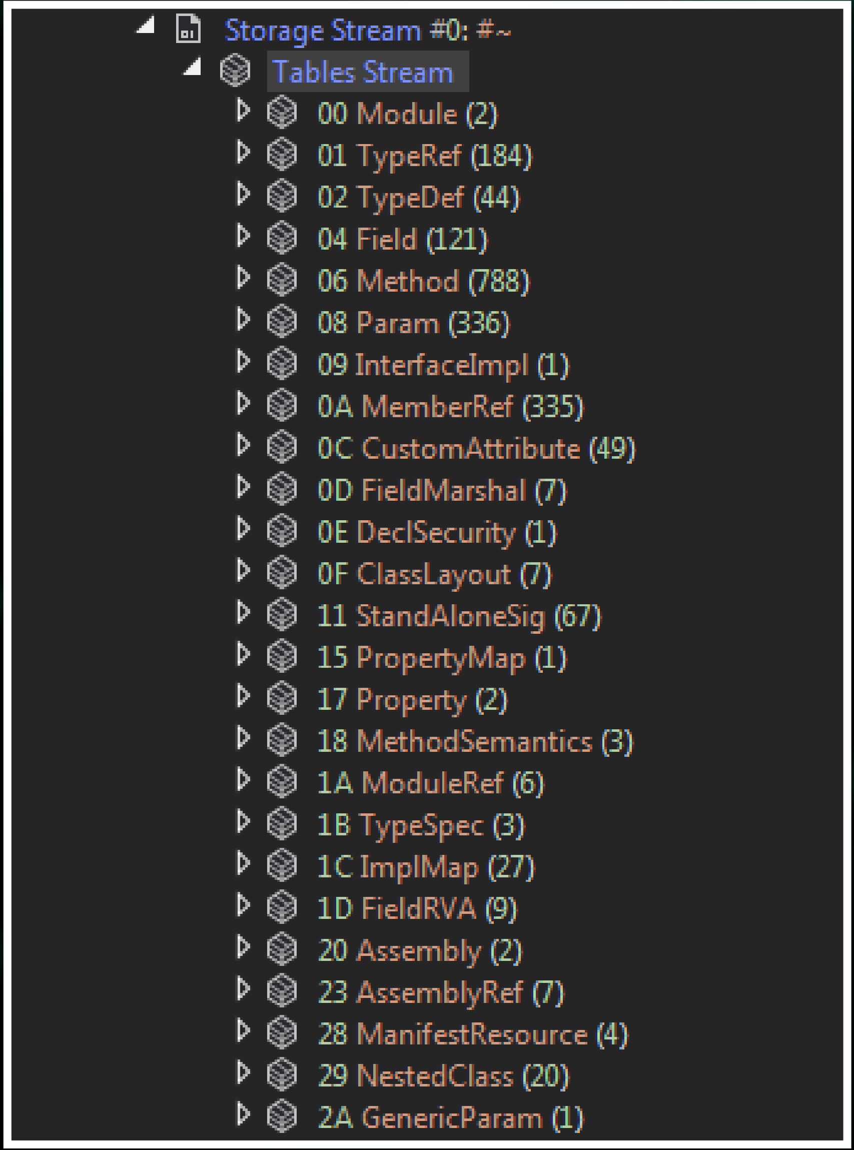
0x00015C70	lSignature	0x424A5342
0x00015C74	iMajorVer	1
0x00015C76	iMinorVer	1
0x00015C78	iExtraData	0
0x00015C7C	iVersionString	0xC
0x00015C80	VersionString	"v2.0.50727\0\0"

From Expert .NET 2.0 IL Assembler by Serge Lidin

What are we looking at?

.NET holds metadata information in a number of streams, each in a different format.

- #Strings: an array of ascii strings, these are referenced by MetaData Tables
- #US: an array of unicode strings, these are referenced directly by code instructions
- #Blob: contains data referenced by MetaData Tables
- #GUID: contains 128 bit long unique identifiers
- #~: contains the MetaData Tables

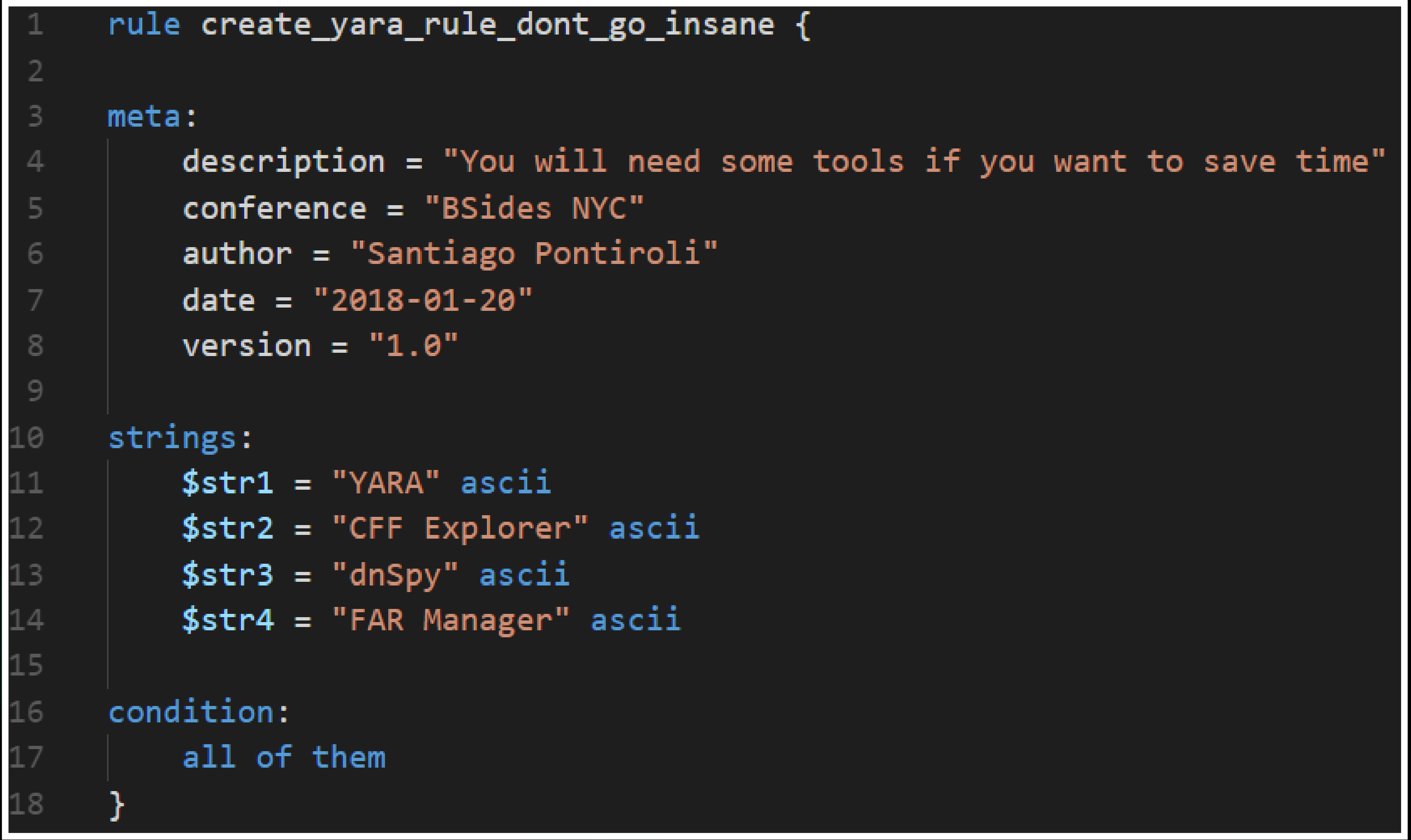


```
Storage Stream #0: #~
  Tables Stream
    00 Module (2)
    01 TypeRef (184)
    02 TypeDef (44)
    04 Field (121)
    06 Method (788)
    08 Param (336)
    09 InterfaceImpl (1)
    0A MemberRef (335)
    0C CustomAttribute (49)
    0D FieldMarshal (7)
    0E DeclSecurity (1)
    0F ClassLayout (7)
    11 StandAloneSig (67)
    15 PropertyMap (1)
    17 Property (2)
    18 MethodSemantics (3)
    1A ModuleRef (6)
    1B TypeSpec (3)
    1C ImplMap (27)
    1D FieldRVA (9)
    20 Assembly (2)
    23 AssemblyRef (7)
    28 ManifestResource (4)
    29 NestedClass (20)
    2A GenericParam (1)
```

http://www.ntcore.com/files/dotnetformat.htm

Let’s use YARA!

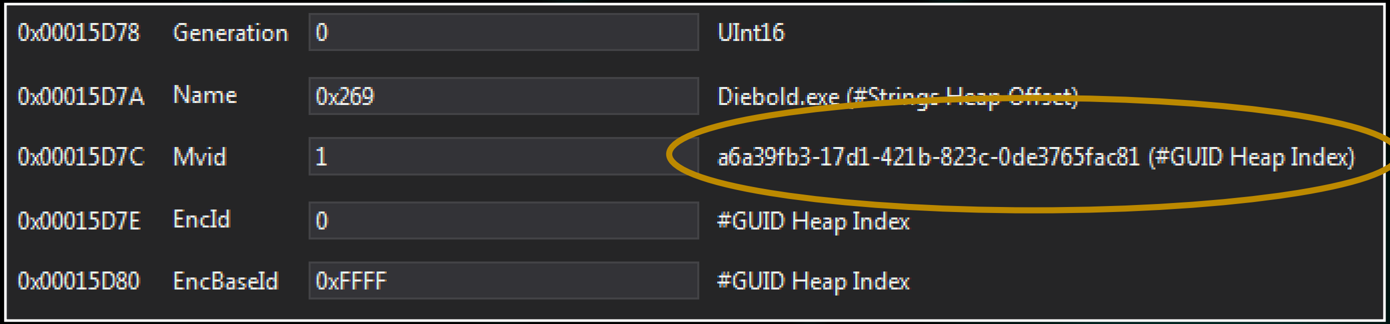
How to write YARA rules for .NET binaries and don’t lose your mind…



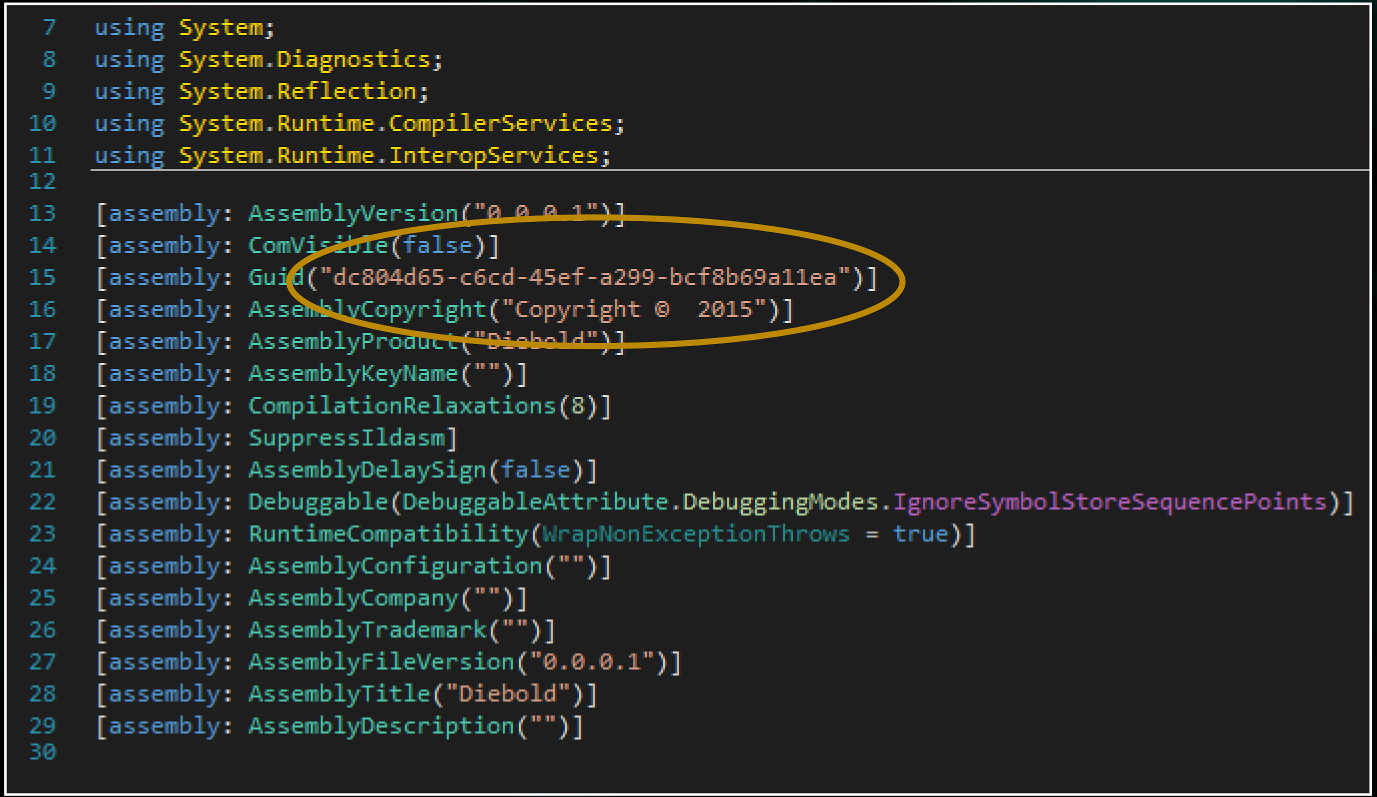
```
rule create_yara_rule_dont_go_insane {

meta:
    description = "You will need some tools if you want to save time"
    conference = "BSides NYC"
    author = "Santiago Pontiroli"
    date = "2018-01-20"
    version = "1.0"

strings:
    $str1 = "YARA" ascii
    $str2 = "CFF Explorer" ascii
    $str3 = "dnSpy" ascii
    $str4 = "FAR Manager" ascii

condition:
    all of them
}
```

Having PE Studio and Visual Studio Code won’t hurt either!

Using GUIDs the right way



0x00015D78	Generation	0	UInt16
0x00015D7A	Name	0x269	Diebold.exe (#Strings Heap Offset)
0x00015D7C	Mvid	1	a6a39fb3-17d1-421b-823c-0de3765fac81 (#GUID Heap Index)
0x00015D7E	EncId	0	#GUID Heap Index
0x00015D80	EncBaseId	0xFFFF	#GUID Heap Index



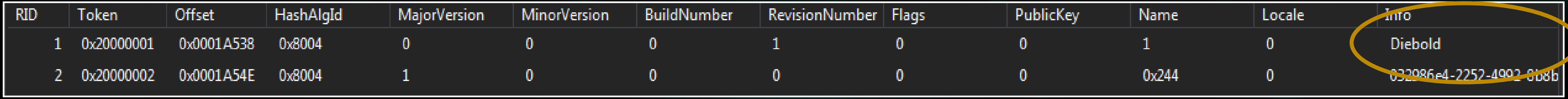
```
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

[assembly: AssemblyVersion("0.0.0.1")]
[assembly: ComVisible(false)]
[assembly: Guid("dc804d65-c6cd-45ef-a299-bcf8b69a11ea")]
[assembly: AssemblyCopyright("Copyright ©  2015")]
[assembly: AssemblyProduct("Diebold")]
[assembly: AssemblyKeyName("")]
[assembly: CompilationRelaxations(8)]
[assembly: SuppressIldasm]
[assembly: AssemblyDelaySign(false)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyFileVersion("0.0.0.1")]
[assembly: AssemblyTitle("Diebold")]
[assembly: AssemblyDescription("")]
```

A TypeLib is GUID but not vice versa.

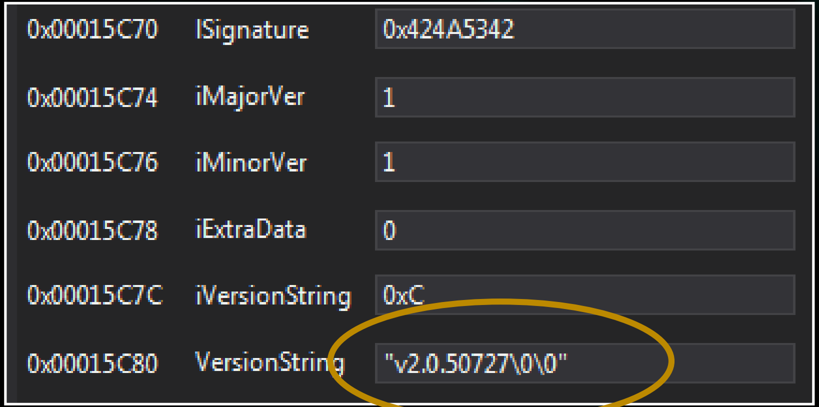
- The **TypeLib** ID is a GUID generated by Visual Studio on the creation of a new project by default.
- The **Module Version ID, or MVID, is a GUID** that can be used to distinguish various versions of a .NET module. This value is generated at build time, resulting in a new GUID for each unique build.

https://www.virusbulletin.com/virusbulletin/2015/06/using-net-guids-help-hunt-malware/

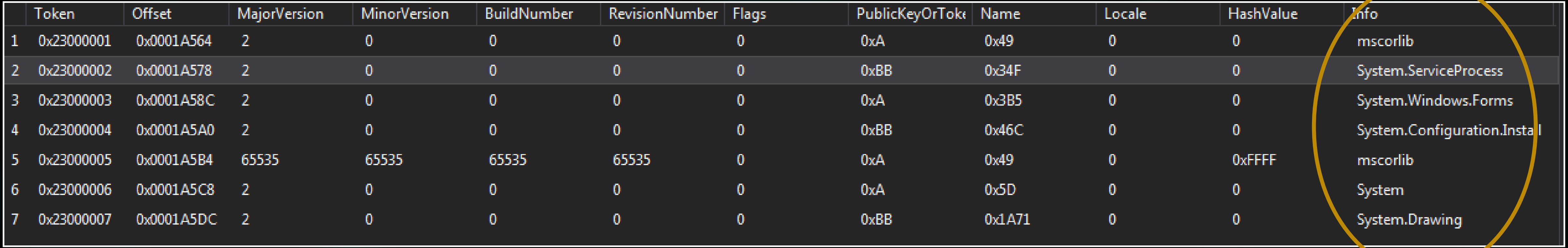
Inspecting the assembly and its references



RID	Token	Offset	HashAlgId	MajorVersion	MinorVersion	BuildNumber	RevisionNumber	Flags	PublicKey	Name	Locale	Info
1	0x20000001	0x0001A538	0x8004	0	0	0	1	0	0	1	0	Diebold
2	0x20000002	0x0001A54E	0x8004	1	0	0	0	0	0	0x244	0	032986e4-2252-4992-8b8b

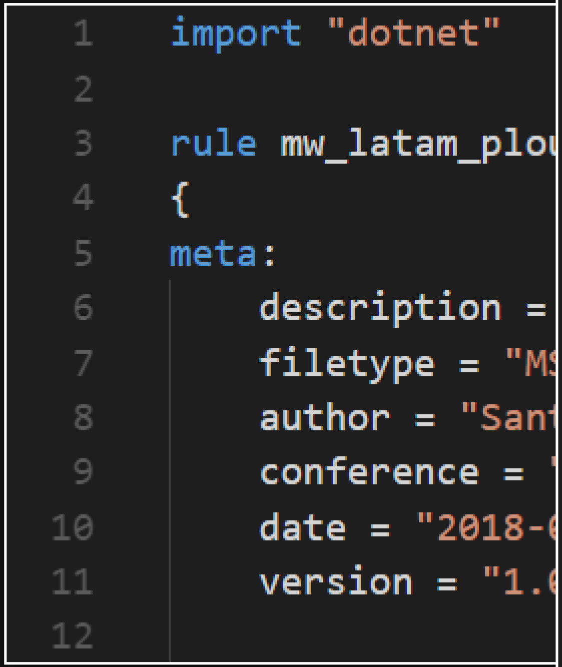


0x00015C70	lSignature	0x424A5342
0x00015C74	iMajorVer	1
0x00015C76	iMinorVer	1
0x00015C78	iExtraData	0
0x00015C7C	iVersionString	0xC
0x00015C80	VersionString	"v2.0.50727\0\0"

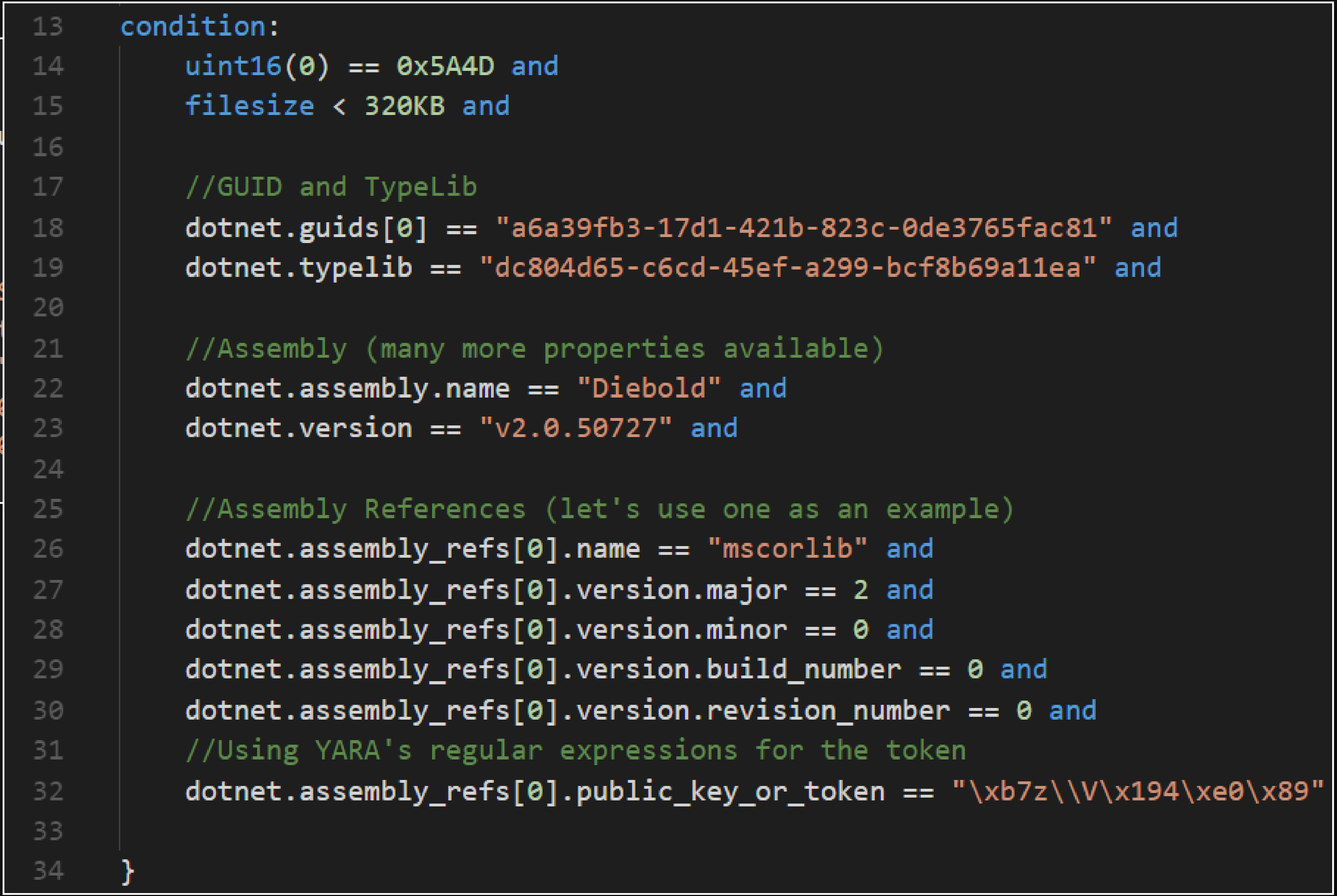


	Token	Offset	MajorVersion	MinorVersion	BuildNumber	RevisionNumber	Flags	PublicKeyOrToke	Name	Locale	HashValue	Info
1	0x23000001	0x0001A564	2	0	0	0	0	0xA	0x49	0	0	mscorlib
2	0x23000002	0x0001A578	2	0	0	0	0	0xBB	0x34F	0	0	System.ServiceProcess
3	0x23000003	0x0001A58C	2	0	0	0	0	0xA	0x3B5	0	0	System.Windows.Forms
4	0x23000004	0x0001A5A0	2	0	0	0	0	0xBB	0x46C	0	0	System.Configuration.Install
5	0x23000005	0x0001A5B4	65535	65535	65535	65535	0	0xA	0x49	0	0xFFFF	mscorlib
6	0x23000006	0x0001A5C8	2	0	0	0	0	0xA	0x5D	0	0	System
7	0x23000007	0x0001A5DC	2	0	0	0	0	0xBB	0x1A71	0	0	System.Drawing

Stop, syntax time!

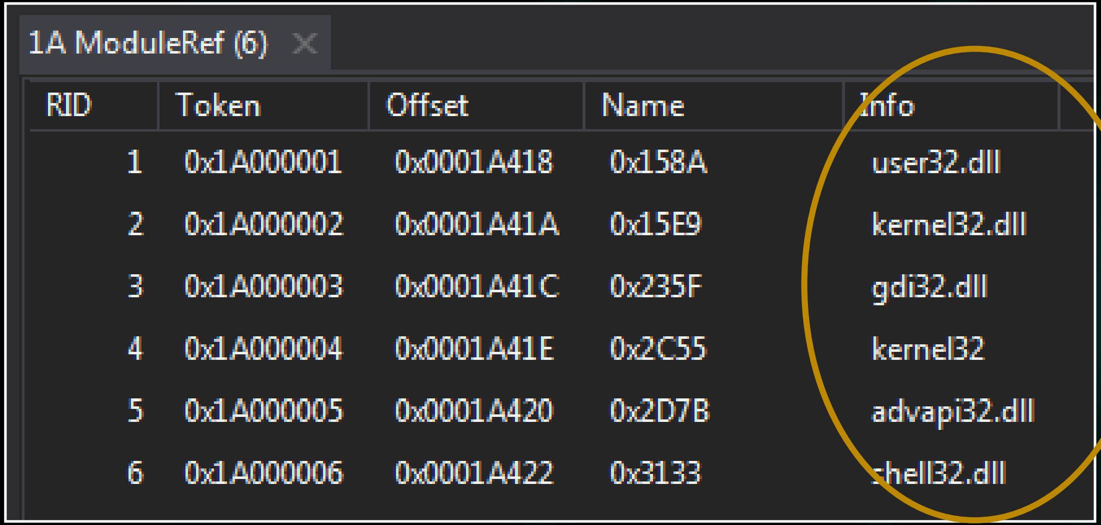


```
import "dotnet"

rule mw_latam_plou
{
meta:
    description =
    filetype = "M
    author = "San
    conference =
    date = "2018-
    version = "1.
```



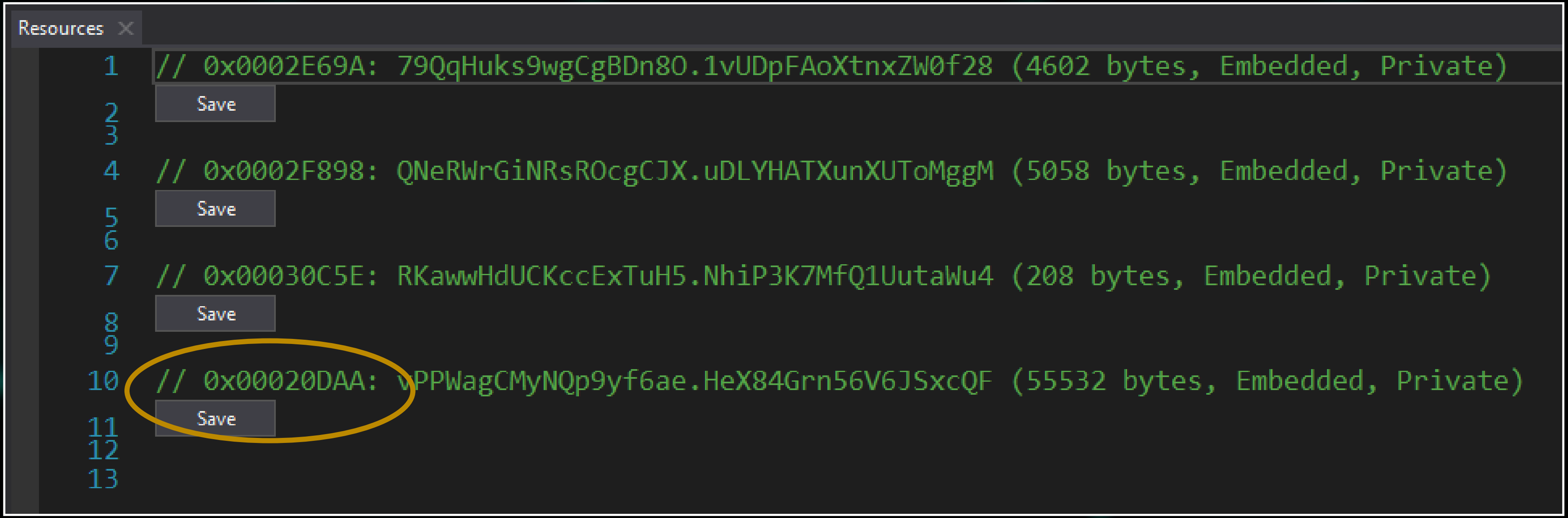
```
condition:
    uint16(0) == 0x5A4D and
    filesize < 320KB and

    //GUID and TypeLib
    dotnet.guids[0] == "a6a39fb3-17d1-421b-823c-0de3765fac81" and
    dotnet.typelib == "dc804d65-c6cd-45ef-a299-bcf8b69a11ea" and

    //Assembly (many more properties available)
    dotnet.assembly.name == "Diebold" and
    dotnet.version == "v2.0.50727" and

    //Assembly References (let's use one as an example)
    dotnet.assembly_refs[0].name == "mscorlib" and
    dotnet.assembly_refs[0].version.major == 2 and
    dotnet.assembly_refs[0].version.minor == 0 and
    dotnet.assembly_refs[0].version.build_number == 0 and
    dotnet.assembly_refs[0].version.revision_number == 0 and
    //Using YARA's regular expressions for the token
    dotnet.assembly_refs[0].public_key_or_token == "\xb7z\\V\x194\xe0\x89"

}
```

Getting more information from the god of wealth, modules and resources



1A ModuleRef (6)

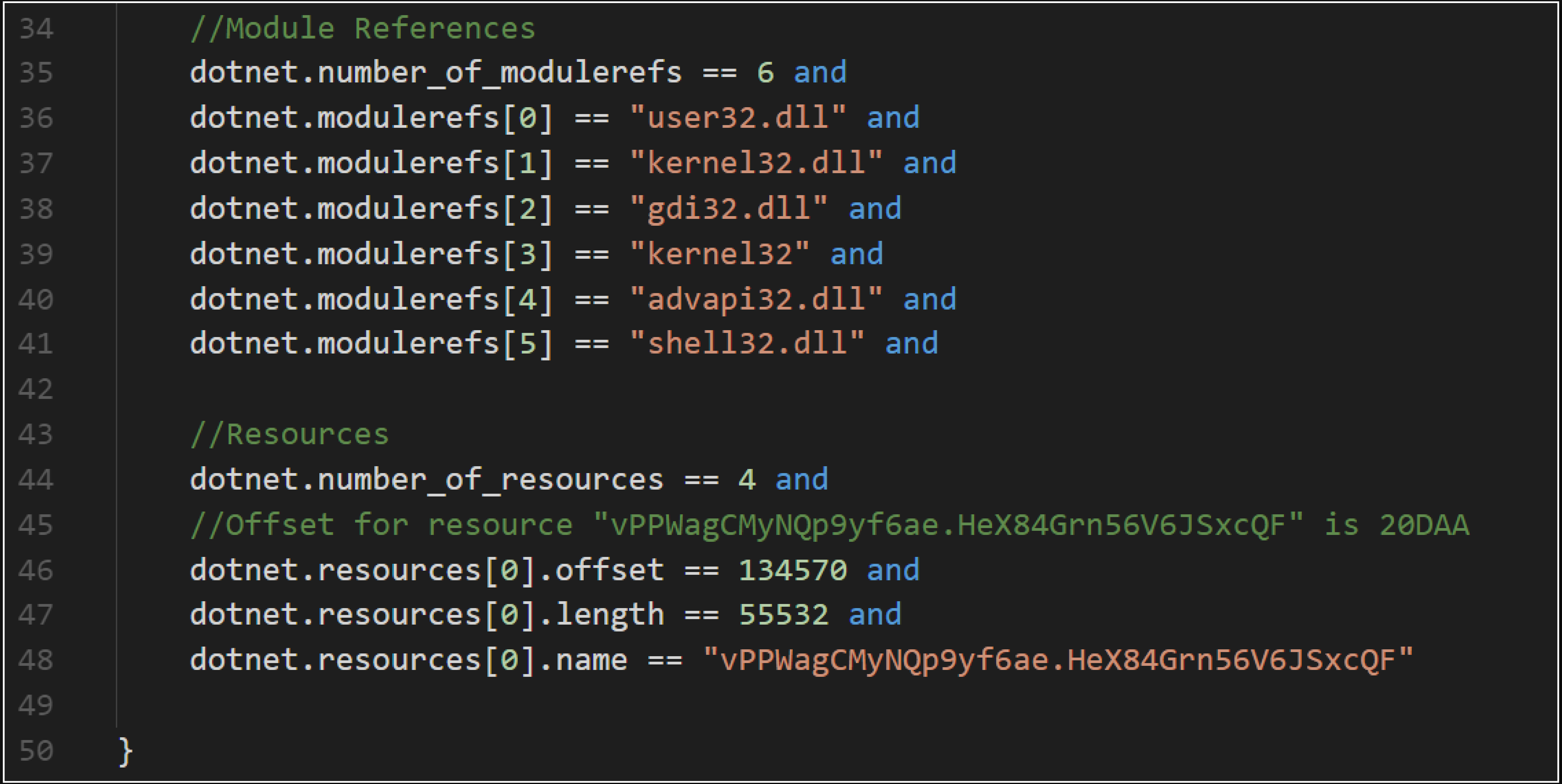
RID	Token	Offset	Name	Info
1	0x1A000001	0x0001A418	0x158A	user32.dll
2	0x1A000002	0x0001A41A	0x15E9	kernel32.dll
3	0x1A000003	0x0001A41C	0x235F	gdi32.dll
4	0x1A000004	0x0001A41E	0x2C55	kernel32
5	0x1A000005	0x0001A420	0x2D7B	advapi32.dll
6	0x1A000006	0x0001A422	0x3133	shell32.dll



Resources

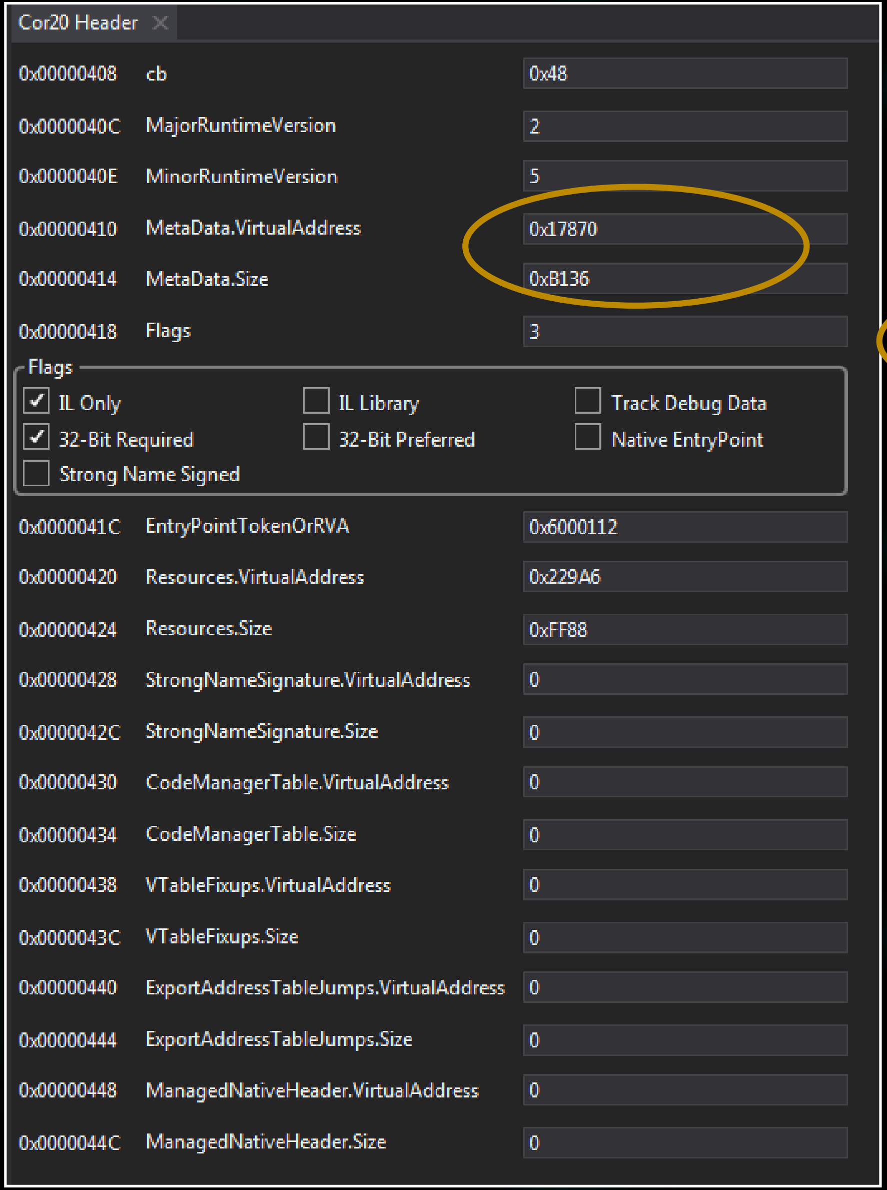
```
// 0x0002E69A: 79QqHuks9wgCgBDn8O.1vUDpFAoXtnxZW0f28 (4602 bytes, Embedded, Private)
Save

// 0x0002F898: QNeRWrGiNRsROcgCJX.uDLYHATXunXUToMggM (5058 bytes, Embedded, Private)
Save

// 0x00030C5E: RKawwHdUCKccExTuH5.NhiP3K7MfQ1UutaWu4 (208 bytes, Embedded, Private)
Save

// 0x00020DAA: vPPWagCMyNQp9yf6ae.HeX84Grn56V6JSxcQF (55532 bytes, Embedded, Private)
Save
```

Stop, syntax time!



```
        //Module References
        dotnet.number_of_modulerefs == 6 and
        dotnet.modulerefs[0] == "user32.dll" and
        dotnet.modulerefs[1] == "kernel32.dll" and
        dotnet.modulerefs[2] == "gdi32.dll" and
        dotnet.modulerefs[3] == "kernel32" and
        dotnet.modulerefs[4] == "advapi32.dll" and
        dotnet.modulerefs[5] == "shell32.dll" and

        //Resources
        dotnet.number_of_resources == 4 and
        //Offset for resource "vPPWagCMyNQp9yf6ae.HeX84Grn56V6JSxcQF" is 20DAA
        dotnet.resources[0].offset == 134570 and
        dotnet.resources[0].length == 55532 and
        dotnet.resources[0].name == "vPPWagCMyNQp9yf6ae.HeX84Grn56V6JSxcQF"

}
```

You must be shapeless, formless, like .NET streams

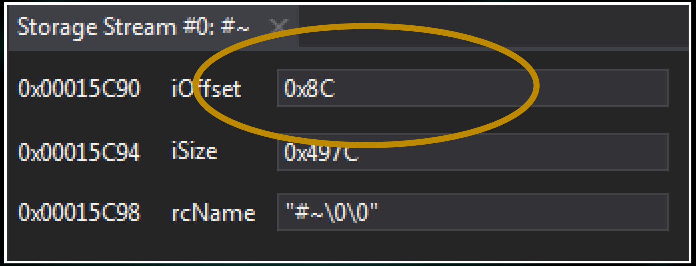


Cor20 Header

0x00000408	cb	0x48
0x0000040C	MajorRuntimeVersion	2
0x0000040E	MinorRuntimeVersion	5
0x00000410	MetaData.VirtualAddress	0x17870
0x00000414	MetaData.Size	0xB136
0x00000418	Flags	3

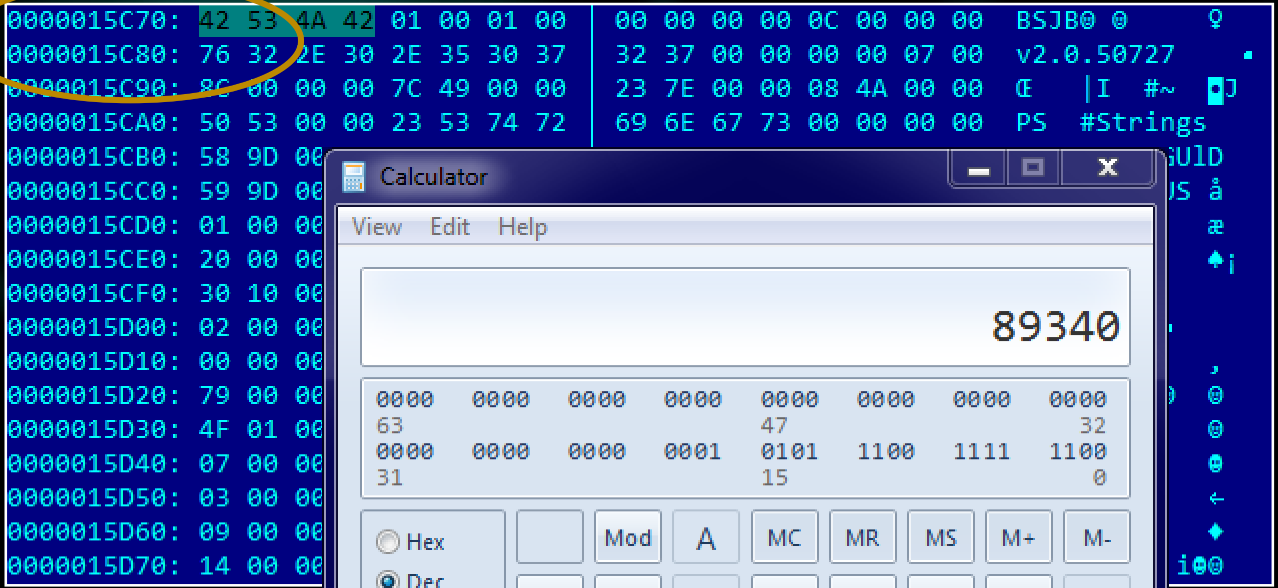
Flags
- [✓] IL Only
- [ ] IL Library
- [ ] Track Debug Data
- [✓] 32-Bit Required
- [ ] 32-Bit Preferred
- [ ] Native EntryPoint
- [ ] Strong Name Signed

0x0000041C	EntryPointTokenOrRVA	0x6000112
0x00000420	Resources.VirtualAddress	0x229A6
0x00000424	Resources.Size	0xFF88
0x00000428	StrongNameSignature.VirtualAddress	0
0x0000042C	StrongNameSignature.Size	0
0x00000430	CodeManagerTable.VirtualAddress	0
0x00000434	CodeManagerTable.Size	0
0x00000438	VTableFixups.VirtualAddress	0
0x0000043C	VTableFixups.Size	0
0x00000440	ExportAddressTableJumps.VirtualAddress	0
0x00000444	ExportAddressTableJumps.Size	0
0x00000448	ManagedNativeHeader.VirtualAddress	0
0x0000044C	ManagedNativeHeader.Size	0

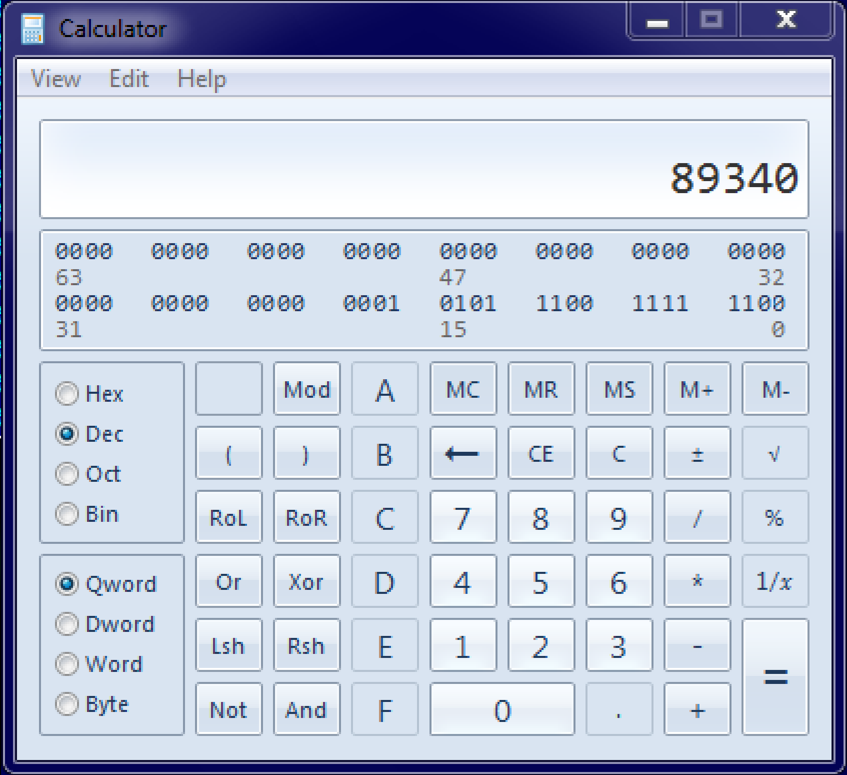


Storage Stream #0: #~

0x00015C90	iOffset	0x8C
0x00015C94	iSize	0x497C
0x00015C98	rcName	"#~\0\0"



```
0000015C70: 42 53 4A 42 01 00 01 00  00 00 00 00 0C 00 00 00  BSJB☻ ☺     ♀
0000015C80: 76 32 2E 30 2E 35 30 37  32 37 00 00 00 00 07 00  v2.0.50727    •
0000015C90: 8C 00 00 00 7C 49 00 00  23 7E 00 00 08 4A 00 00  Œ   |I  #~  ◘J
0000015CA0: 50 53 00 00 23 53 74 72  69 6E 67 73 00 00 00 00  PS  #Strings
0000015CB0: 58 9D 00                                           UlD
0000015CC0: 59 9D 00                                           US å
0000015CD0: 01 00 00                                              æ
0000015CE0: 20 00 00                                              ♠¡
0000015CF0: 30 10 00
0000015D00: 02 00 00
0000015D10: 00 00 00
0000015D20: 79 00 00                                              ☺
0000015D30: 4F 01 00                                              ☺
0000015D40: 07 00 00                                              ☻
0000015D50: 03 00 00                                              ←
0000015D60: 09 00 00                                              ♦
0000015D70: 14 00 00                                             i☻☺
```



Calculator

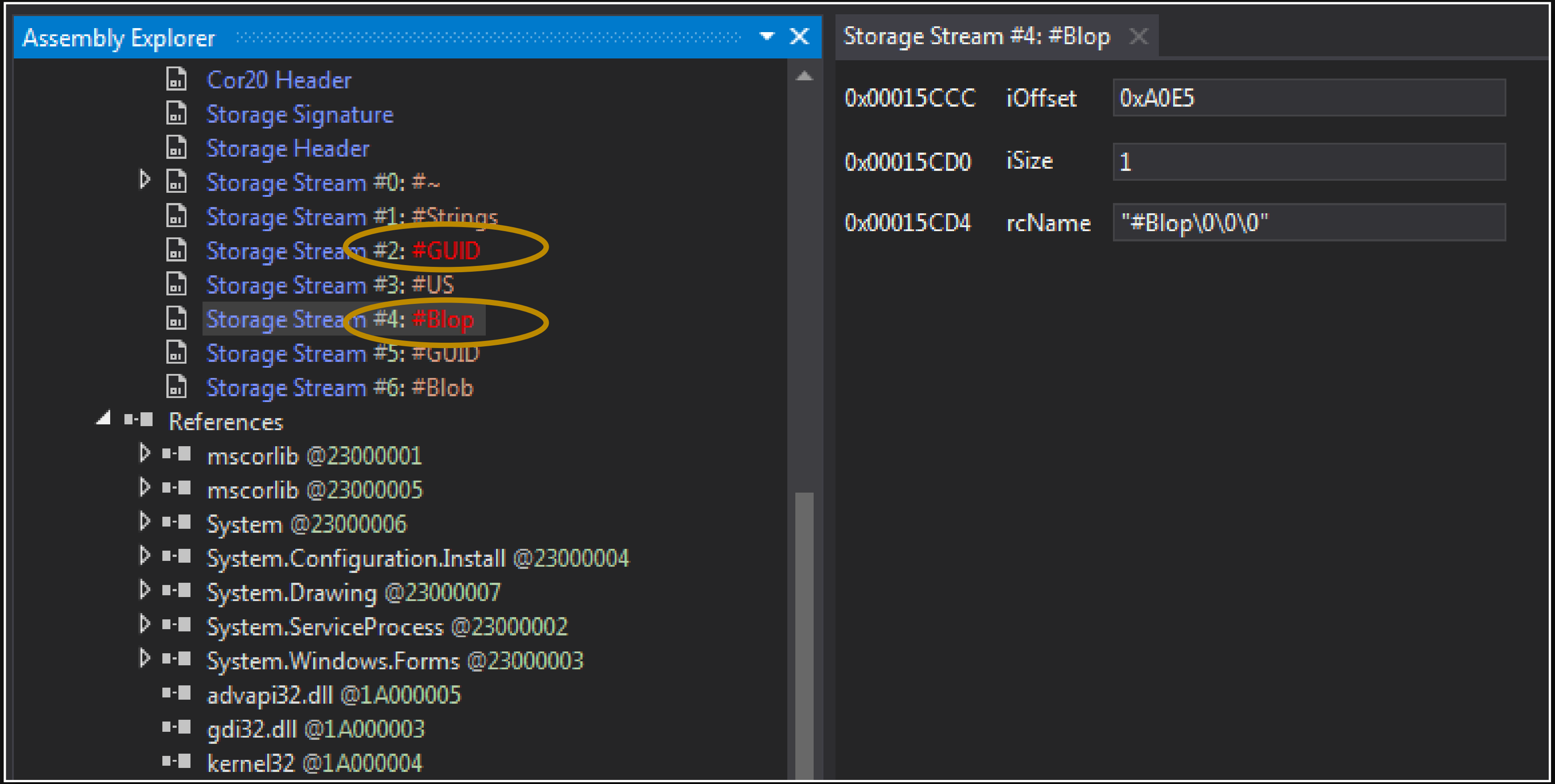
```
89340

0000   0000   0000   0000   0000   0000   0000   0000
63                          47                     32
0000   0000   0000   0001   0101   1100   1111   1100
31                          15                      0
```

[✓ Dec] Hex Oct Bin

[✓ Qword] Dword Word Byte

KASPERSKY lab

Not all streams are created equal



Assembly Explorer

- Cor20 Header
- Storage Signature
- Storage Header
- Storage Stream #0: #~
- Storage Stream #1: #Strings
- Storage Stream #2: #GUID
- Storage Stream #3: #US
- Storage Stream #4: #Blop
- Storage Stream #5: #GUID
- Storage Stream #6: #Blob
- References
  - mscorlib @23000001
  - mscorlib @23000005
  - System @23000006
  - System.Configuration.Install @23000004
  - System.Drawing @23000007
  - System.ServiceProcess @23000002
  - System.Windows.Forms @23000003
  - advapi32.dll @1A000005
  - gdi32.dll @1A000003
  - kernel32 @1A000004

Storage Stream #4: #Blop

0x00015CCC	iOffset	0xA0E5
0x00015CD0	iSize	1
0x00015CD4	rcName	"#Blop\0\0\0"

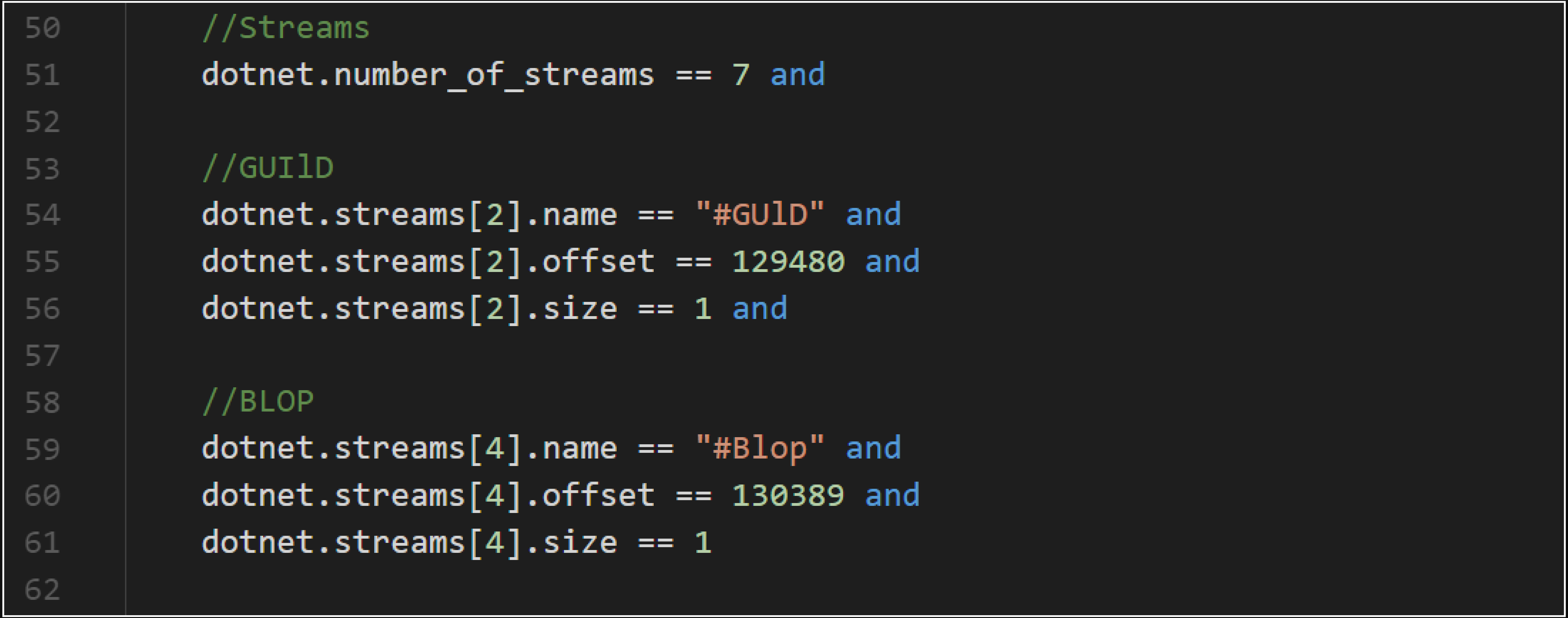
Finding interesting indicators in the binary

GUID != GUlD

BLOB != BLOP

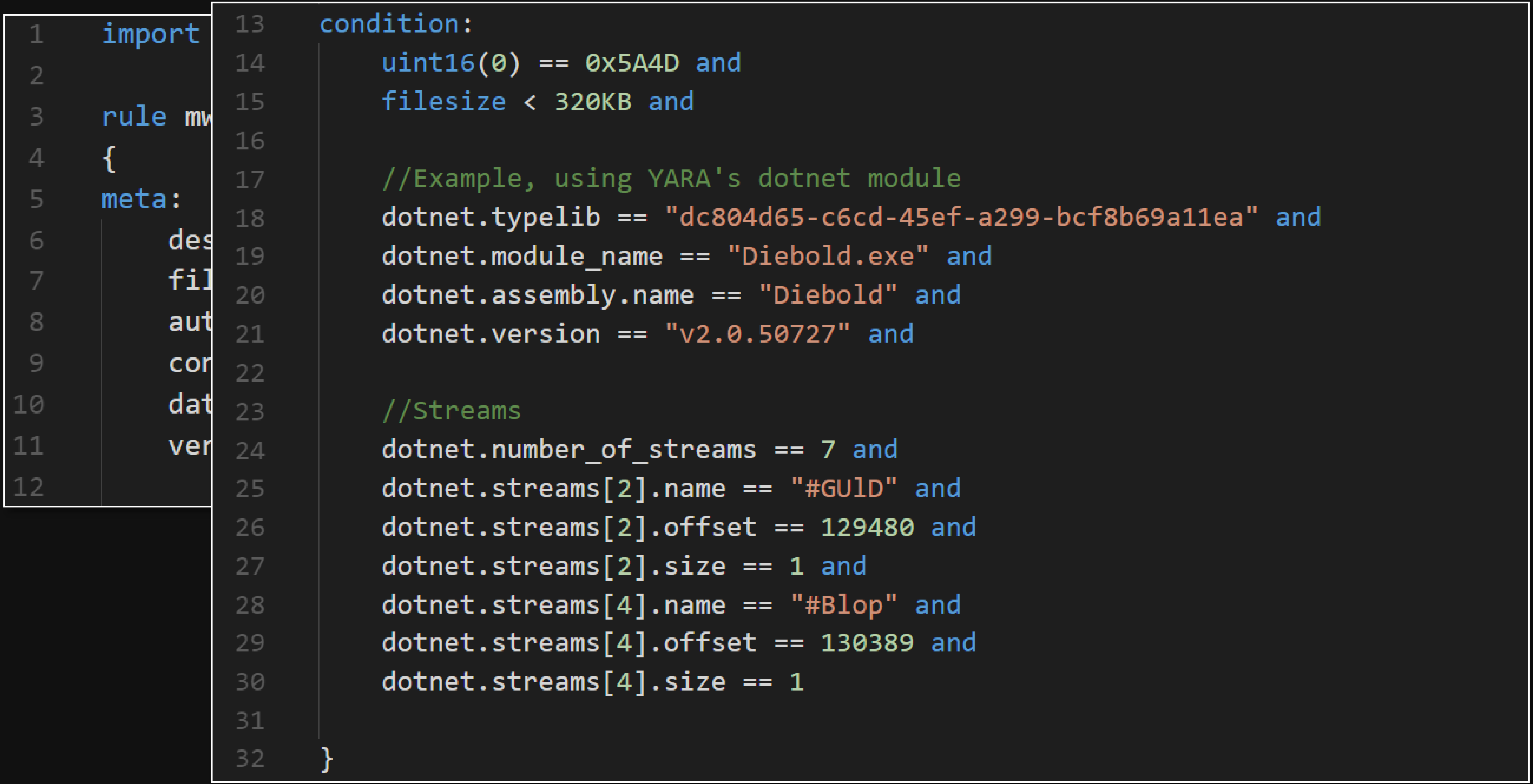
KASPERSKY lab

Stop, syntax time!



```
    //Streams
    dotnet.number_of_streams == 7 and

    //GUIlD
    dotnet.streams[2].name == "#GUlD" and
    dotnet.streams[2].offset == 129480 and
    dotnet.streams[2].size == 1 and

    //BLOP
    dotnet.streams[4].name == "#Blop" and
    dotnet.streams[4].offset == 130389 and
    dotnet.streams[4].size == 1

```

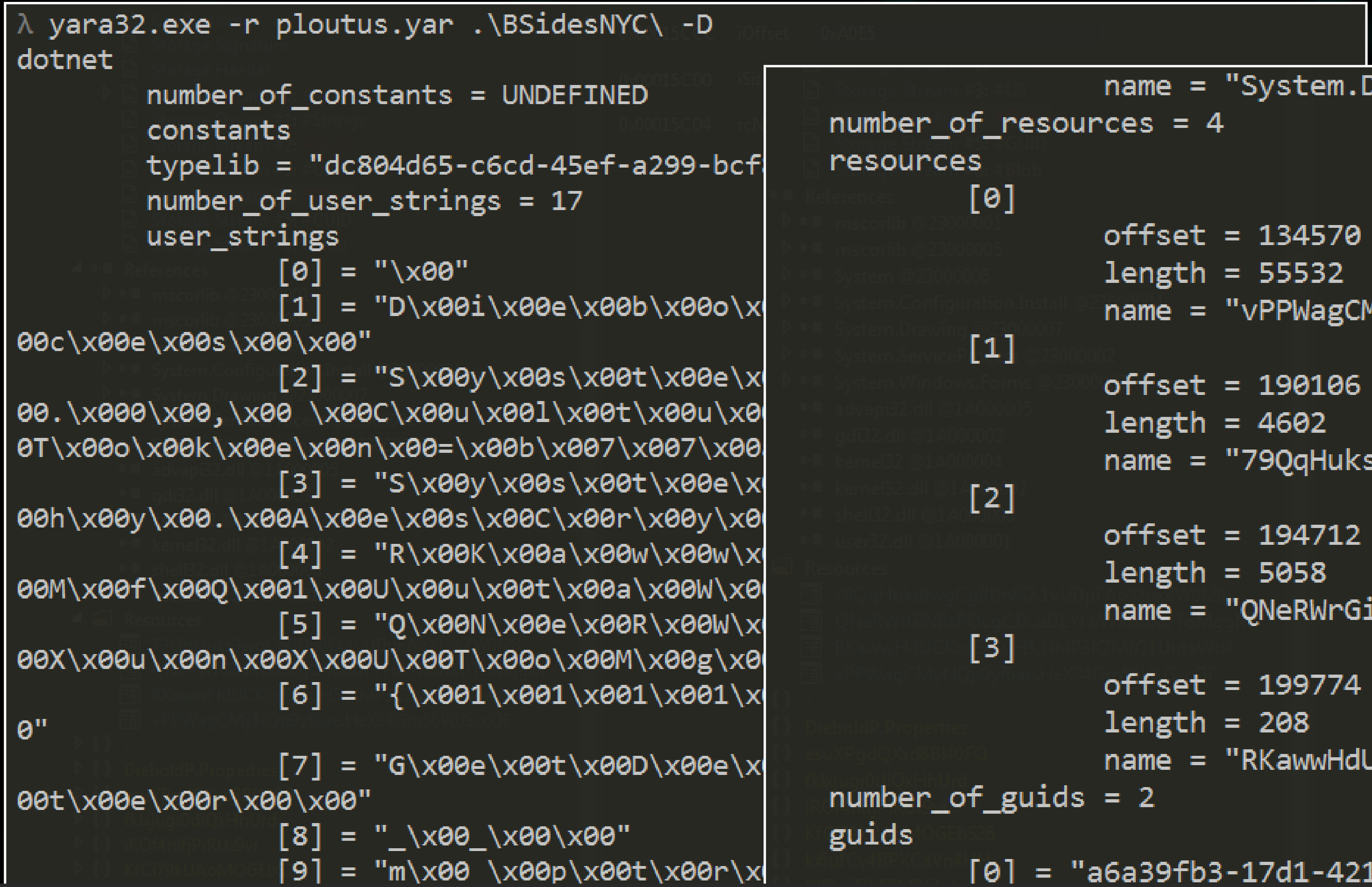
Let’s build our “final” YARA rule



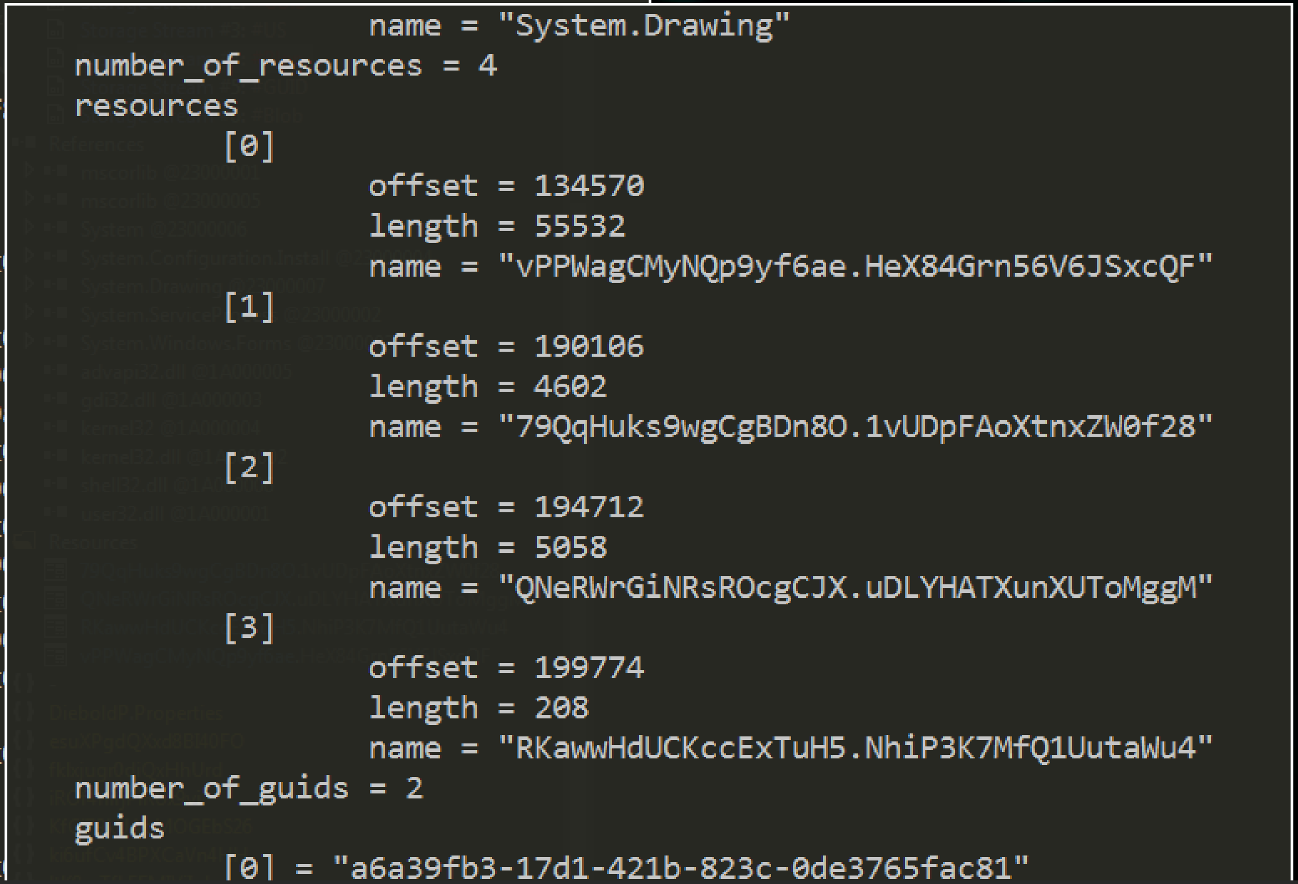
```
import

rule mw
{
meta:
    des
    fil
    aut
    cor
    dat
    ver

```

```
condition:
    uint16(0) == 0x5A4D and
    filesize < 320KB and

    //Example, using YARA's dotnet module
    dotnet.typelib == "dc804d65-c6cd-45ef-a299-bcf8b69a11ea" and
    dotnet.module_name == "Diebold.exe" and
    dotnet.assembly.name == "Diebold" and
    dotnet.version == "v2.0.50727" and

    //Streams
    dotnet.number_of_streams == 7 and
    dotnet.streams[2].name == "#GUlD" and
    dotnet.streams[2].offset == 129480 and
    dotnet.streams[2].size == 1 and
    dotnet.streams[4].name == "#Blop" and
    dotnet.streams[4].offset == 130389 and
    dotnet.streams[4].size == 1

}
```

It sounds like too much work, but YARA always likes to help



```
λ yara32.exe -r ploutus.yar .\BSidesNYC\ -D
dotnet
        number_of_constants = UNDEFINED
        constants
        typelib = "dc804d65-c6cd-45ef-a299-bcf
        number_of_user_strings = 17
        user_strings
                [0] = "\x00"
                [1] = "D\x00i\x00e\x00b\x00o\x
00c\x00e\x00s\x00\x00"
                [2] = "S\x00y\x00s\x00t\x00e\x
00.\x000\x00,\x00 \x00C\x00u\x00l\x00t\x00u\x0
0T\x00o\x00k\x00e\x00n\x00=\x00b\x007\x007\x00
                [3] = "S\x00y\x00s\x00t\x00e\x
00h\x00y\x00.\x00A\x00e\x00s\x00C\x00r\x00y\x0
                [4] = "R\x00K\x00a\x00w\x00w\x
00M\x00f\x00Q\x001\x00U\x00u\x00t\x00a\x00W\x0
                [5] = "Q\x00N\x00e\x00R\x00W\x
00X\x00u\x00n\x00X\x00U\x00T\x00o\x00M\x00g\x0
                [6] = "{\x001\x001\x001\x001\x
0"
                [7] = "G\x00e\x00t\x00D\x00e\x
00t\x00e\x00r\x00\x00"
                [8] = "_\x00_\x00\x00"
                [9] = "m\x00 \x00p\x00t\x00r\x
```

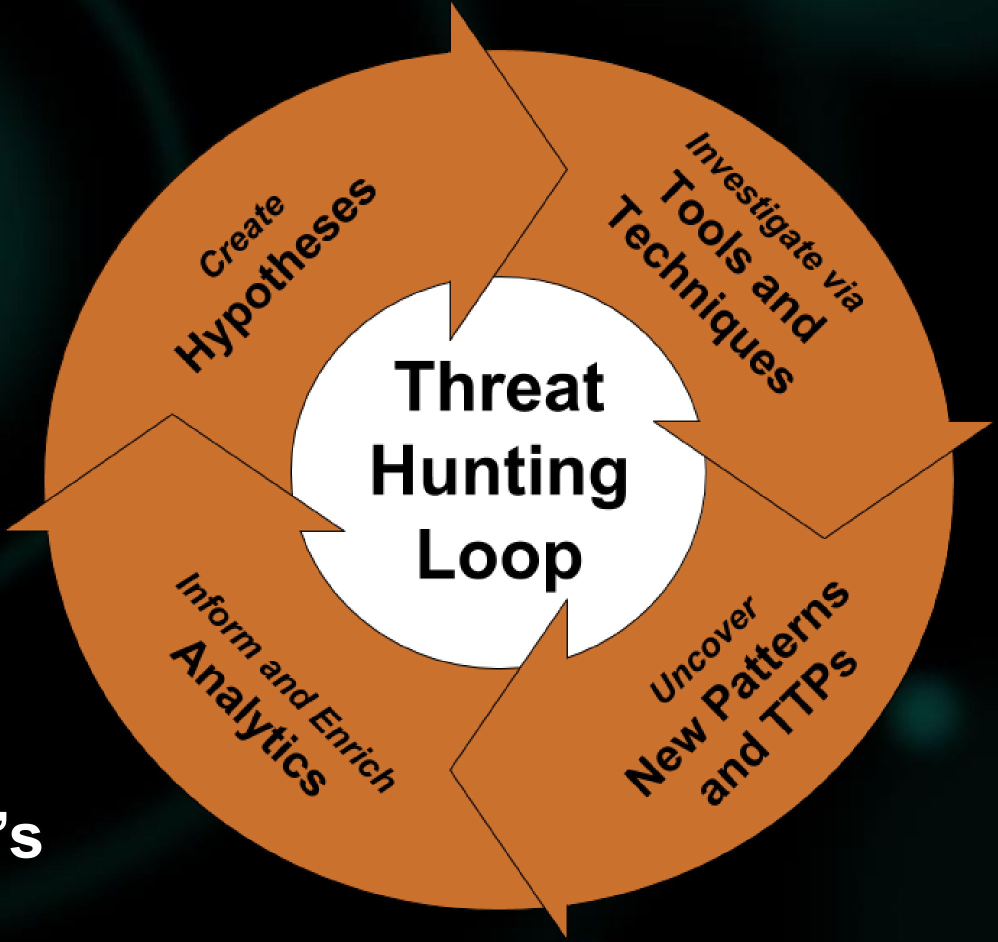


```
                        name = "System.Drawing"
    number_of_resources = 4
    resources
            [0]
                        offset = 134570
                        length = 55532
                        name = "vPPWagCMyNQp9yf6ae.HeX84Grn56V6JSxcQF"
            [1]
                        offset = 190106
                        length = 4602
                        name = "79QqHuks9wgCgBDn8O.1vUDpFAoXtnxZW0f28"
            [2]
                        offset = 194712
                        length = 5058
                        name = "QNeRWrGiNRsROcgCJX.uDLYHATXunXUToMggM"
            [3]
                        offset = 199774
                        length = 208
                        name = "RKawwHdUCKccExTuH5.NhiP3K7MfQ1UutaWu4"
    number_of_guids = 2
    guids
            [0] = "a6a39fb3-17d1-421b-823c-0de3765fac81"
```

The D flag will help you debug your rules,
yara32.exe -r ploutus.yar .\BSidesNYC\ -D

Do you really need all that information?

- Are you trying to find new samples of a known threat or something similar and still unknown?
- What is the lowest amount of information you can use to identify the sample?
- If it only detects that sample, the rule is rarely useful (except incident response cases).
- If it detects too much and the number of false positives hinders your investigation it's time to go back to the drawing board.


Threat Hunting Loop
Create Hypotheses
Investigate via Tools and Techniques
Uncover New Patterns and TTPs
Inform and Enrich Analytics

Additional resources, becoming a YARA ninja

- How to write simple but sound YARA rules (Florian Roth)
  - https://www.bsk-consulting.de/2015/02/16/write-simple-sound-yara-rules/
- Using .NET GUIDs to help hunt for malware (Brian Wallace)
  - https://www.virusbulletin.com/virusbulletin/2015/06/using-net-guids-help-hunt-malware
- YARA dotnet module documentation
  - https://yara.readthedocs.io/en/v3.7.0/modules/dotnet.html
- YARA dotnet module source code
  - https://github.com/VirusTotal/yara/blob/master/libyara/modules/dotnet.c

A special thank you goes to **Wesley Shields (@wxs)** and **Victor Alvarez (@plusivc)**

Thank you! Questions?

Santiago Martin Pontiroli

Global Research and Analysis Team

Kaspersky Lab

@spontiroli

KASPERSKY lab