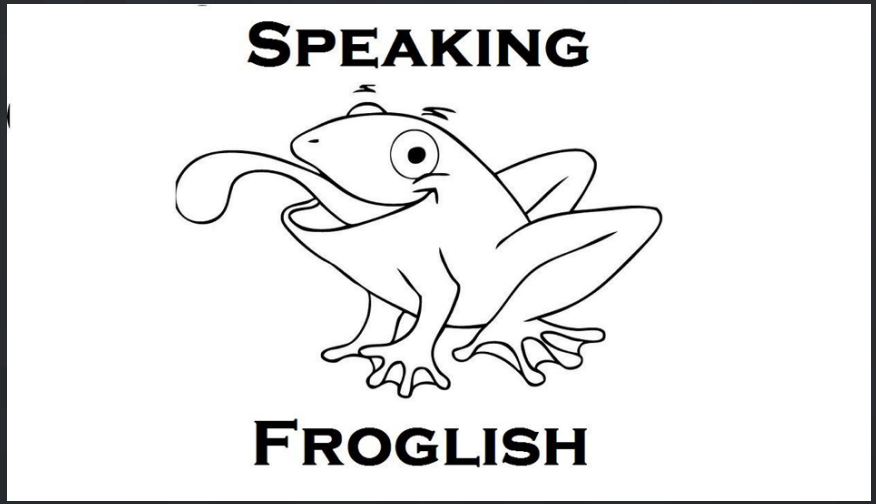
Let's Play With WinDBG and .NET

Paul Rascagneres
Senior Threat Researcher

whoami

- Paul Rascagneres – prascagn@cisco.com
- Security Researcher at Talos Outreach
- Malware & APT hunter for more than 7 years…
- Co-Organizer of Botconf



TALOS

Agenda

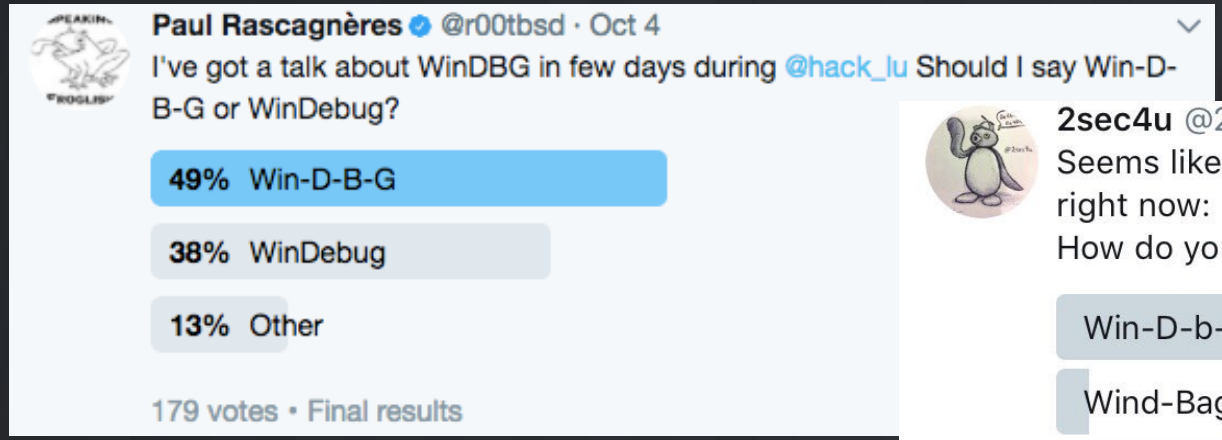
- WinDBG Introduction
- Case Study 1: PowerShell Analysis
- Case Study 2: .NET Unpacker
- PYKD Script
- Conclusion

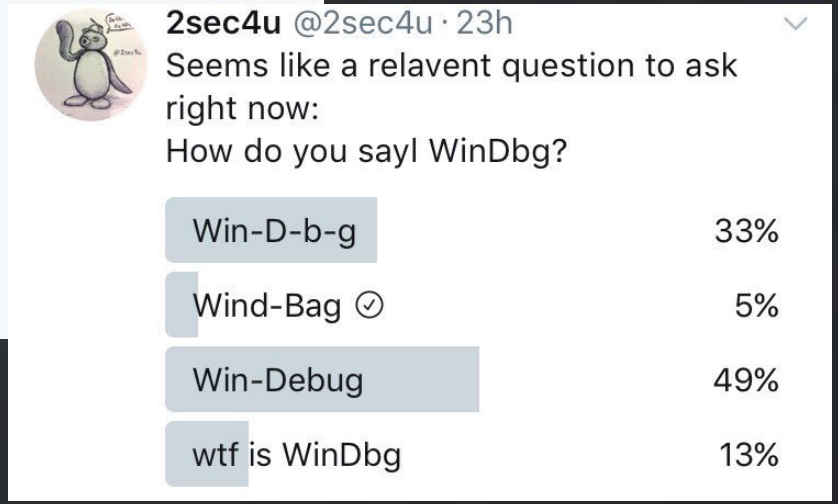
TALOS

WinDBG Introduction

WinDBG Introduction

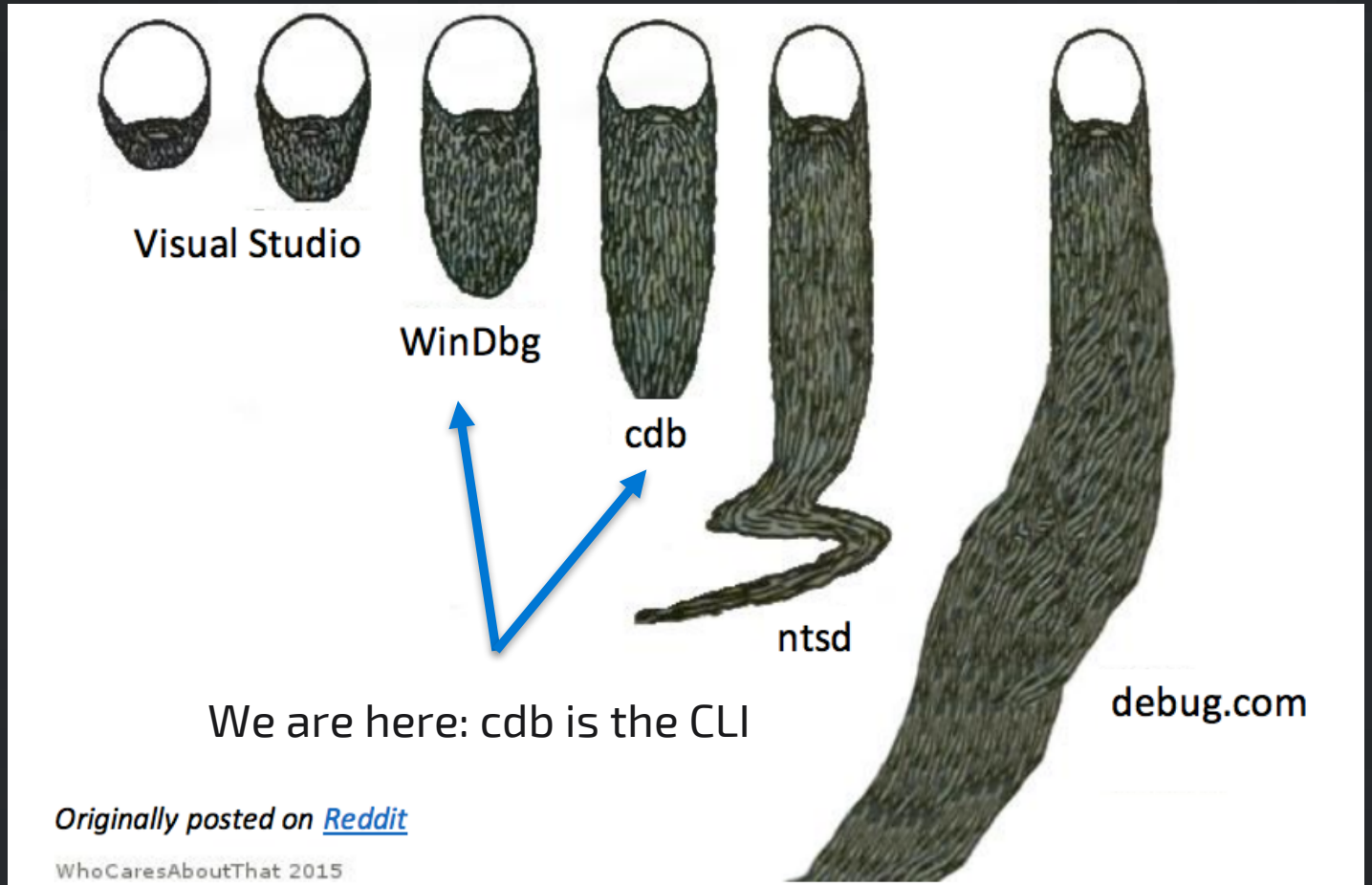
- Naming convention





TALOS

WinDBG Introduction



TALOS

WinDBG Introduction

- Free Microsoft Debugger: https://msdn.microsoft.com/en-us/library/windows/hardware/ff551063(v=vs.85).aspx

- Mainly used for unmanaged code analysis…
- … but perfectly support .NET debugging

- Why is it not popular?
  - Archaic user interface
  - Unintuitive syntax
  - Extremely unintuitive scripting language

- But extremely powerful with some practice.

TALOS

WinDBG Introduction

- **Unmanaged** code analysis

- Display memory content: dX (where X is the type of data to display)
  Syntax: *dX start_address* [*end_address*|L*size*]
  - db: display bytes
  - dp: display pointers
  - du: display unicode
- Example:

```
0:000> db 029e30f0
029e30f0  41 42 43 44 45 00 00 00-00 00 00 00 00 00 00 00  ABCDE...........
029e3100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
029e3160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

TALOS

WinDBG Introduction

- **Unmanaged** code analysis

- Breakpoint
  - bp address : breakpoint at the address (or function)
  - bl: list breakpoints
  - bd X: disable the breakpoint X
  - be X: enable the breakpoint X
  - g: Go, execute the binary until the next breakpoint

- Breakpoint on specific event
  - sxe ld *library*: break when *library* is loaded

TALOS

WinDBG Introduction

- **Unmanaged** code analysis
- Dump memory content in a file
  - .writemem filename start_address “end_address|Lsize”
- List loaded modules: lm
- List exported functions: x
- Useful documentation: http://windbg.info/doc/1-common-cmds.html

TALOS

WinDBG Introduction

- **Managed** code analysis
- First step: load sos extension
  - .loadby sos clr
- SOS provides a new sets of commands dedicated to .NET debugging
- Help: !help

TALOS

WinDBG Introduction

- **Managed** code analysis

- Breakpoint: !bpmd
  - !bpmd library functionname: set a breakpoint
  - !bpmd –list: list the current breakpoints
  - !bpmd -clearall: remove the current breakpoints
- Example:

```
0:000> !bpmd mscorlib.dll System.Reflection.Assembly.Load
Found 8 methods in module 71041000...
MethodDesc = 71100b50
MethodDesc = 71100b7c
[redacted]
Setting breakpoint: bp 71B29095 [System.Reflection.Assembly.Load(Byte[], Byte[],
System.Security.Policy.Evidence)]
Setting breakpoint: bp 71B29037 [System.Reflection.Assembly.Load(Byte[], Byte[],
System.Security.SecurityContextSource)]
[redacted]
Adding pending breakpoints...
```

TALOS

WinDBG Introduction

- **Managed** code analysis
- Get information on the Stack (managed code only): !CLRStack
- Example:

```
0:008> !CLRStack -p
OS Thread Id: 0x2d34 (8)
        Child SP               IP Call Site
000000a7f9ace700 00007fff977c96d9
System.Diagnostics.Process.Start(System.Diagnostics.ProcessStartInfo)
    PARAMETERS:
        startInfo (<CLR reg>) = 0x0000028cbd5faa18
```

TALOS

WinDBG Introduction

- **Managed** code analysis
- Display information about an object (managed code only): !DumpObj
- Example:

```
0:009> !DumpObj /d 0000016580593b18
Name:        System.String
MethodTable: 00007fff9897de98
EEClass:     00007fff982d35f0
Size:        82(0x52) bytes
File:        C:\WINDOWS\Microsoft.Net\\v4.0_4.0.0.0__b77a5c561934e089\mscorlib.dll
String:      blaba
Fields:
              MT    Field   Offset                 Type VT     Attr            Value Name
00007fff989807d8  400026f        8          System.Int32  1 instance               28
m_stringLength
00007fff9897f050  4000270        c           System.Char  1 instance               68
m_firstChar
00007fff9897de98  4000274       90         System.String  0   shared         static Empty
```

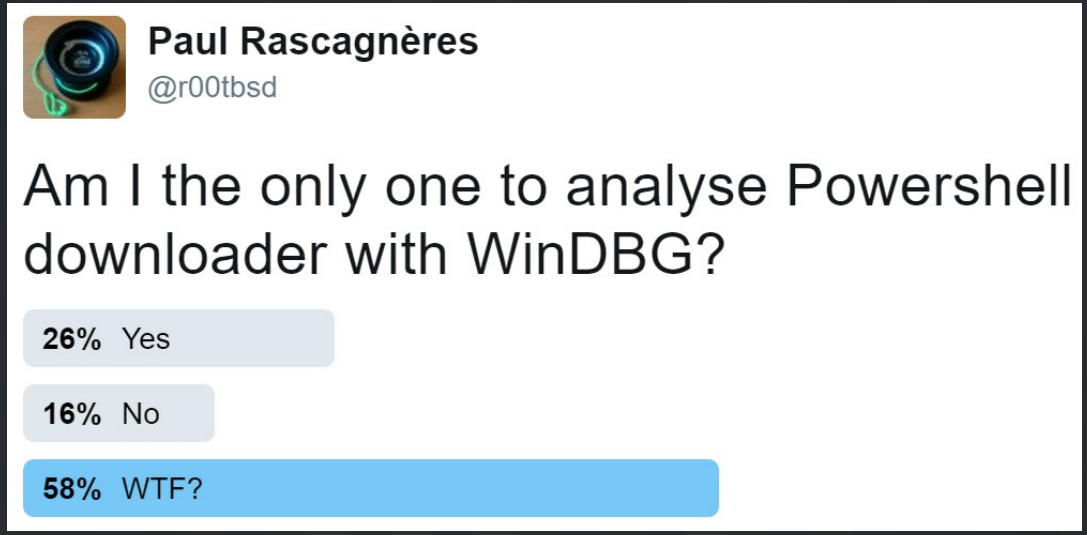
WinDBG Introduction

- **Managed** code analysis
- Complete documentation of SOS commands: https://msdn.microsoft.com/en-us/library/bb190764(v=vs.110).aspx

TALOS

Case Study 1: PowerShell Analysis

Case Study 1: PowerShell Analysis



TALOS

Case Study 1: PowerShell Analysis

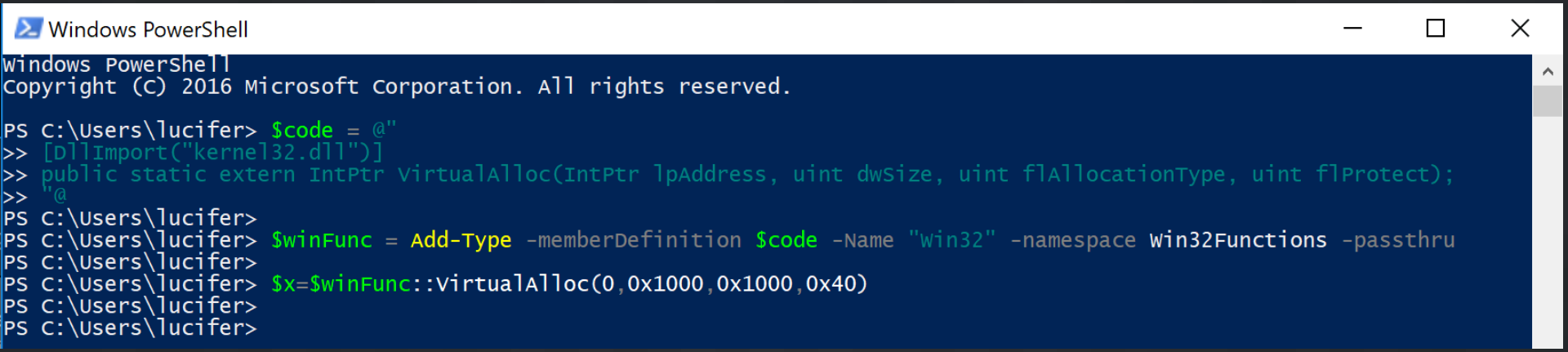
- Powershell is an unmissable tool for malware developers…
- How to automate Powershell analysis ?
- Can we debug Powershell scripts with WinDBG ?

YES we can

TALOS

Case Study 1: PowerShell Analysis

- Usage of unmanaged code (for example dllimport)



- Standard WinDBG breakpoint => bp kernelbase!VirtualAlloc
- No specific WinDBG tricks, debug “as usual”

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
PS C:\Users\lucifer> start-process notepad.exe
```

- SOS for .NET analysis + breakpoint

```
0:011> .loadby sos clr

0:011> !bpmd system.dll System.Diagnostics.Process.Start
Found 6 methods in module 00007fff97581000...
breakpoint: bp 00007FFF977C96D9 [System.Diagnostics.Process.Start(System.
breakpoint: bp 00007FFF97E8057D [System.Diagnostics.Process.Start(System.
breakpoint: bp 00007FFF97E80539 [System.Diagnostics.Process.Start(System.
breakpoint: bp 00007FFF97E804B6 [System.Diagnostics.Process.Start(System.
breakpoint: bp 00007FFF97E80436 [System.Diagnostics.Process.Start(System.
System.String, breakpoint: bp 00007FFF977C72DA [System.Diagnostics.Proces
Adding pending breakpoints...
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
PS C:\Users\lucifer> start-process notepad.exe
```

- .NET breakpoint & arguments playing

```
Breakpoint 0 hit
System_ni+0x2496d9:
00007fff`977c96d9 488d0d08711e00  lea     rcx,[System_ni+0x4307e8
(00007fff`979b07e8)]
0:008> !CLRStack -p
OS Thread Id: 0x2d34 (8)
        Child SP               IP Call Site
000000a7f9ace700 00007fff977c96d9
System.Diagnostics.Process.Start(System.Diagnostics.ProcessStartInfo)
    PARAMETERS:
        startInfo (<CLR reg>) = 0x0000028cbd5faa18
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
PS C:\Users\lucifer> start-process notepad.exe
```

- .NET breakpoint & arguments playing

```
0:008> !DumpObj /d 0000028cbd5faa18
Name:        System.Diagnostics.ProcessStartInfo
MethodTable: 00007fff979ae380
EEClass:     00007fff975e29f0
Size:        144(0x90) bytes
File:
C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll
Fields:
              MT    Field   Offset                 Type VT     Attr            Value Name
00007fff9897de98  40027f3        8        System.String  0 instance 0000028cbd5fde18 fileName
00007fff9897de98  40027f4       10        System.String  0 instance 0000000000000000 arguments
[...redacted...]
00007fff9897ad70  4002806       58 System.WeakReference  0 instance 0000000000 weakParentProces
00007fff979af0a0  4002807       60 ....StringDictionary  0 instance 000000 environmentVariables
00007fff982e5ec0  4002808       68 ...tring, mscorlib]]  0 instance 0000000000000 environment
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
PS C:\Users\lucifer> start-process notepad.exe
```

- .NET breakpoint & arguments playing

```
0:008> !DumpObj /d 0000028cbd5fde18
Name:        System.String
MethodTable: 00007fff9897de98
EEClass:     00007fff982d35f0
Size:        88(0x58) bytes
File:
C:\WINDOWS\Microsoft.Net\assembly\GAC_64\mscorlib\v4.0_4.0.0.0__b77a5c56
1934e089\mscorlib.dll
String:      C:\WINDOWS\system32\notepad.exe
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
PS C:\Users\lucifer> start-process notepad.exe
```

- For geeks directly in RCX

```
0:008> dp rcx+8 L1
0000028c`bd5faa20  0000028c`bd5fde18
0:008> du 0000028c`bd5fde18+0xC
0000028c`bd5fde24  "C:\WINDOWS\system32\notepad.exe"
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
$a = New-Object System.Net.WebClient
$a.DownloadFile("http://blog.talosintelligence.com/", "c:\users\lucifer\desktop\demo.txt")
```

- SOS for .NET analysis + breakpoint

```
0:011> .loadby sos clr

0:008> !bpmd system.dll System.Net.WebClient.DownloadFile
Found 2 methods in module 00007fff97581000...
MethodDesc = 00007fff976c1fe8
MethodDesc = 00007fff976c1ff8
Setting breakpoint: bp 00007FFF97DCAE0C
[System.Net.WebClient.DownloadFile(System.Uri, System.String)]
Setting breakpoint: bp 00007FFF97DCADBC
[System.Net.WebClient.DownloadFile(System.String, System.String)]
Adding pending breakpoints...
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
$a = New-Object System.Net.WebClient
$a.DownloadFile("http://blog.talosintelligence.com/", "c:\users\lucifer\desktop\demo.txt")
```

- SOS for .NET analysis + breakpoint

```
Breakpoint 7 hit
System_ni+0x84adbc:
00007fff`97dcadbc 4885d2          test    rdx,rdx
```

TALOS

Case Study 1: PowerShell Analysis

- Usage of managed code == .NET framework

```
$a = New-Object System.Net.WebClient
$a.DownloadFile("http://blog.talosintelligence.com/", "c:\users\lucifer\desktop\demo.txt")
```

- SOS for .NET analysis + breakpoint

```
0:008> du rdx+c
0000028c`bd53f13c  "http://blog.talosintelligence.co"
0000028c`bd53f17c  "m/"
0:008> du r8+c
0000028c`bd53f3b4  "c:\users\lucifer\desktop\demo.tx"
0000028c`bd53f3f4  "t"
```

TALOS

Case Study 1: PowerShell Analysis

- Fully automate the process:

```
PS C:\Users\lucifer>
PS C:\Users\lucifer> $a = New-Object System.Net.WebClient
PS C:\Users\lucifer> $a.DownloadFile("http://blog.talosintelligence.com/", "c:\users\lucifer\desktop\demo.txt")
PS C:\Users\lucifer> start-process "c:\users\lucifer\desktop\demo.txt"
PS C:\Users\lucifer>
```

```
0:020> bp 00007FFF977C96D9 ".echo 'Executed command: ';r $t1=poi(rcx+8);du $t1+0xC;g"
breakpoint 0 redefined
0:020> bp 00007FFF97DCADBC ".echo 'Argument 1: ';du rdx+c;.echo 'Argument 2: ';du r8+c;g"
breakpoint 7 redefined
0:020> g
'Argument 1: '
0000028c`bd53236c  "http://blog.talosintelligence.co"
0000028c`bd5323ac  "m/"
'Argument 2: '
0000028c`bd53255c  "c:\users\lucifer\desktop\demo.tx"
0000028c`bd53259c  "t"
'Executed command: '
0000028c`bd5962e4  "C:\users\lucifer\desktop\demo.tx"
0000028c`bd596324  "t"
```

TALOS

Case Study 2: .NET Unpacker

Case Study 2: .NET Unpacker

- First step: analysis of the packer with ILSpy: http://ilspy.net/
- Sample: 45c695e610d78178ec5ca6f4e1993afacf4e435b566cd2caf65408fb6080300f

```
// Global type: <Module>
// Entry point: ob6eaGgG7Bht6B35c0.G9puOotvCiNCkEEPD9.XHh0nc9pu
// Architecture: x86
// Runtime: .N
```

```
G9puOotvCiNCkEEPD9.VwJCBSgIt = "ა지ს있료ი나는을모운모님ლ1Ha6QLh5qEpS6XBK19aN9EoC24XEHx7cwiu5qmVHfYpCT";
G9puOotvCiNCkEEPD9.DnAZg86lO = "드지6ქ이여난예생지를돈행난ი435L3Ct7dWJ87R6R4Jx3O4tlkI6Jv";
G9puOotvCiNCkEEPD9.Sx9cvJDpn = "모모그님6이종ვ이ა료님종ა는S1P1qWxq8s2Z8tOj0T5TSqzFNTEBx";
G9puOotvCiNCkEEPD9.sXGwopsMo = G9puOotvCiNCkEEPD9.vovYCiNCk(Convert.FromBase64String(G9puOotvCiNCkEEPD9.EHQI8XHAH), "도그것ა것수ცნდ새დოㅅ하예음ი난하도ო하님것დ");
G9puOotvCiNCkEEPD9.dGKEtiads = "ოდ그종하코ი을로수ა도하하난8l9DiTJ9dqqnx6UGZS0MeL5xQSSFA746ZiZOZGexBSmP8qH";
G9puOotvCiNCkEEPD9.NOiVUMHB3 = "ქ코새დ을수종음하კიდ를ᄅ코R97LB52QI17xmMWMe587Bk9Q2kHPxn7H3Qq3x7Y6MyyFFYiL1p6Ic3KK6F";
G9puOotvCiNCkEEPD9.hMoHRDR8b = "დქ난다ჯლდ을고ა를여을ა6UudN70439R0WL46iVGCU52D8CBf17zRjP9nV52W10xl09mfa4Lso3CP8";
arg_12A_0 = 6;
continue;
IL_FA:
G9puOotvCiNCkEEPD9.EHQI8XHAH = "YRHUW2RYZ1hnVGtUa1VrVGuco5yjnKOco1xjXGNcY1xjXGNcY1xjXGNcY1xjXGNcY1xjXGNcY9zj3OPO0jP+wUpshpjvz6TO0Js8mjLtYutl81oEztGLU/5T/U5
rwM5BjkFJ+XJw+ZbXGNcX75wHA8c+11iXVjpvoG+fS6fb0t0281lV2hXZ0VHZyUEvW5Rbk3Q3q59LulvUG9GnBAvEC/rrIS7hLF7IRiZ77pUQHkwddjU5tl2M+SWqZafireHsneuHwczDMPmsUl2SWw9BDqlbktg38j
sYUyCxMqz+dMcSUYVWhDfAI9qY0hHPLLhLL2x9fmZFLh3KKdWl9aZRkmz/DJ9oi3DigWKdXqRXFNchwhezsDPNLrIRUpFujXIhSJtDoFaFXC+4O8tYEUK+Pa/rja45ukoZX1yObY49JDfPLN5dA0CV1iSWyTqkR5dEl
ByYENgk1CRkhGSYYJf29e0R5RV5P75emk6SXoowyuHpFc0RyRF5cUe15Pvivj5ufoJ2ZnZmgn56ZjqGWpZqkk6SXqJajlKeYpourjpGthKoU1uHd4tXq1erV6reHuIa5MgseyoNb9H3TfvxLEcmGRRG/Etl+z4FHBTk
G9puOotvCiNCkEEPD9.hyykEIZPn = "새수수ლა도ბდ종하ი여프ა2C15rO4K73o8L0Q7RdjboqnlX4Ttk2Pw3";
G9puOotvCiNCkEEPD9.sGbxdnxP8 = "6르료ბ모돈도하시ო예ქ만하დj0S1H1LN175M8PKEp3N1BpyY5b2dKHf6N4yIQwOT1o1";
goto IL_33;
IL_DA:
G9puOotvCiNCkEEPD9.iHSpObpyl = "을하세종것를ა음ი나도ცსუდ하8mh437GuTsDNw83uFw8n1TY8l2W80Rdl1wuHb9KO0njjuR7Tr0cOo1U3";
G9puOotvCiNCkEEPD9.SLp9TaZBv = "ϑ을지그დი하ბ3만하ბ하스0HZl5yt8178a16v7K11HN3ZI8iyccdV1517WOj166";
```

TALOS

Case Study 2: .NET Unpacker

- First step: analysis of the packer with ILSpy: http://ilspy.net/
- Sample: 45c695e610d78178ec5ca6f4e1993afacf4e435b566cd2caf65408fb6080300f

```
tr[(int)(num3 % unchecked((long)ptr.Length))] = (byte)(((int)((ptr[(int)(num3 % unchecked((long)ptr.Length))] ^ array[(int)(num3 % unchecked((long)array.Length))])
g_42_0 = 5;
 (!G9puOotvCiNCkEEPD9.QRrHSOvbpyl1LpTaZB.A732HMPrQZlUIkEbd4())
```

```
case 10:
    instance = Assembly.Load(G9puOotvCiNCkEEPD9.sXGwopsMo);
    goto IL_1B7;
```

TALOS

Case Study 2: .NET Unpacker

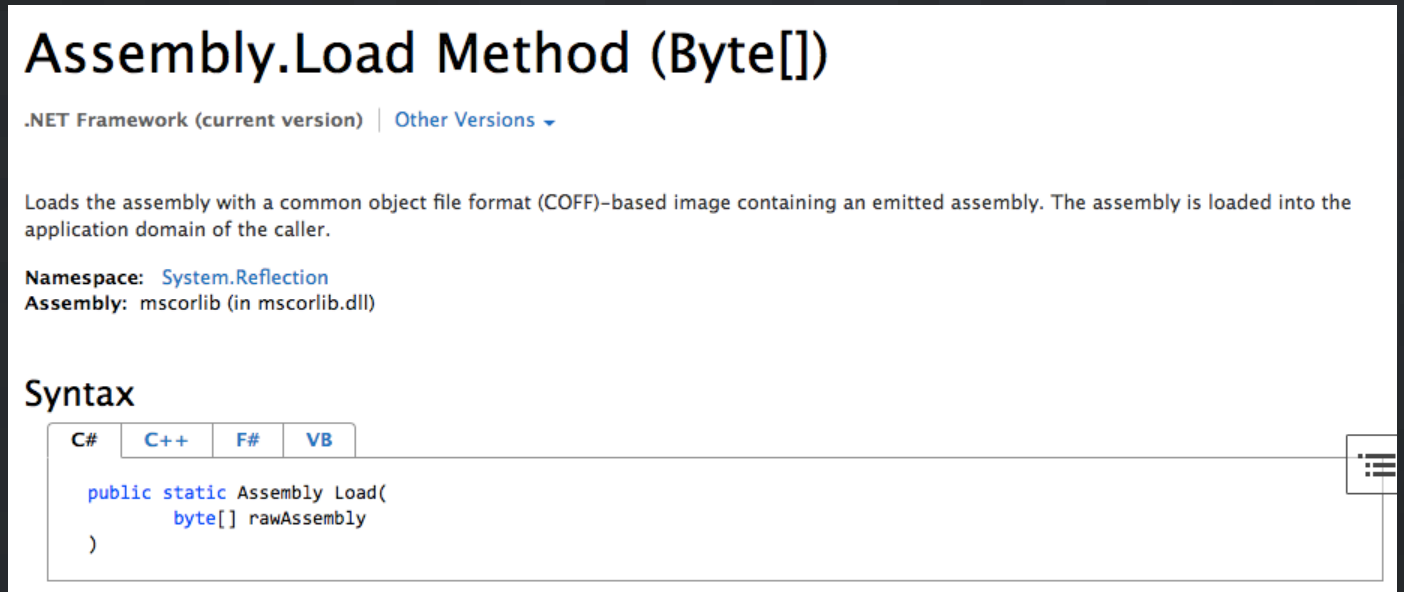
- First step: analysis of the packer with ILSpy: http://ilspy.net/
- Same packer but different algorithm: AES

```
RijndaelManaged rijndaelManaged = new RijndaelManaged();
IL_2F:
IL_41:
checked
{
    byte[] array3;
    try
    {
        rijndaelManaged.KeySize = 256;
        lgerIx6IxIuc71DOAH.dFS7APUNaV1lPZDfkm.rQ9DVZfwfWcdLpuvqv(rijndaelManaged, lgerIx6IxIuc71DOAH.dFS7APUNaV1lPZDfkm.BmVAoMoVh(text));
        rijndaelManaged.IV = lgerIx6IxIuc71DOAH.dFS7APUNaV1lPZDfkm.cHEIfjLPZKJjLDG3u8(text);
        MemoryStream memoryStream = new MemoryStream(array);
        try
        {
            CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateDecryptor(), CryptoStreamMode.Read);
            try
            {
                byte[] array2 = new byte[array.Length + 1];
                lgerIx6IxIuc71DOAH.dFS7APUNaV1lPZDfkm.oJyJM5nS9SCp6XGCSB();
                int arg_FF_0 = lgerIx6IxIuc71DOAH.dFS7APUNaV1lPZDfkm.iNOHipSpRkgKh5Phpd() ? 2 : 4;
                while (true)
                {
                    int num;
                    switch (arg_FF_0)
                    {
                    case 0:
                    case 4:
                        num = cryptoStream.Read(array2, 0, array.Length);
```

TALOS

Case Study 2: .NET Unpacker

- First step: analysis of the packer with ILSpy: http://ilspy.net/
- Common point:
  - Assembly.Load() API



Assembly.Load Method (Byte[])

.NET Framework (current version) | Other Versions

Loads the assembly with a common object file format (COFF)-based image containing an emitted assembly. The assembly is loaded into the application domain of the caller.

Namespace: System.Reflection
Assembly: mscorlib (in mscorlib.dll)

Syntax

C# | C++ | F# | VB

```
public static Assembly Load(
        byte[] rawAssembly
)
```

TALOS

Case Study 2: .NET Unpacker

- Second step: WinDBG debugging
- Execute the malware and load the necessary modules

```
0:000> sxe ld clrjit
0:000> g
(1e88.1c88): Unknown exception - code 04242420 (first chance)
ModLoad: 71950000 719d0000
C:\Windows\Microsoft.NET\Framework\v4.0.30319\clrjit.dll
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=0094e000
eip=77e21fec esp=00afe704 ebp=00afe758 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
ntdll!NtMapViewOfSection+0xc:
77e21fec c22800          ret     28h
0:000> .loadby sos clr
```

TALOS

Case Study 2: .NET Unpacker

- Second step: WinDBG debugging
- Execute the malware and breakpoint on Assembly.Load()

```
0:000> !bpmd mscorlib.dll System.Reflection.Assembly.Load
Found 8 methods in module 719d1000...
[redacted]
MethodDesc = 71a95740
MethodDesc = 71a95764
[redacted]
Setting breakpoint: bp 7251C71C [System.Reflection.Assembly.Load(Byte[])]
Setting breakpoint: bp 71D4921D
[System.Reflection.Assembly.Load(System.Reflection.AssemblyName,
System.Security.Policy.Evidence)]
[redacted]
Adding pending breakpoints...
```

TALOS

Case Study 2: .NET Unpacker

- Second step: WinDBG debugging
- Execute the malware and breakpoint on Assembly.Load()

```
0:000> g
Breakpoint 3 hit
eax=00000000 ebx=00aff4bc ecx=02b57e04 edx=00000000 esi=02b57e04 edi=00aff430
eip=7251c71c esp=00aff400 ebp=00aff408 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
mscorlib_ni+0xb4c71c:
7251c71c e87f8efcff      call    mscorlib_ni+0xb155a0 (724e55a0)
0:000> !CLRStack -p
OS Thread Id: 0x1c88 (0)
Child SP       IP Call Site
00aff400 7251c71c System.Reflection.Assembly.Load(Byte[])
    PARAMETERS:
        rawAssembly (<CLR reg>) = 0x02b57e04
```

TALOS

Case Study 2: .NET Unpacker

- Second step: WinDBG debugging
- Execute the malware and breakpoint on Assembly.Load()

```
0:000> !DumpObj /d 02b57e04
Name:        System.Byte[]
MethodTable: 71ecb264.   EEClass:     71a6e304
Size:        5644(0x160c) bytes
Array:       Rank 1, Number of elements 5632, Type Byte (Print Array)
Content:
MZ.....................@.....................................!..L.!Th
is program cannot be run in DOS mode....$.......

0:000> db 0x02b57e04
02b57e04  64 b2 ec 71 00 16 00 00-4d 5a 90 00 03 00 00 00  d..q....MZ......
02b57e14  04 00 00 00 ff ff 00 00-b8 00 00 00 00 00 00 00  ................
02b57e24  40 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  @...............
02b57e34  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
02b57e44  00 00 00 00 80 00 00 00-0e 1f ba 0e 00 b4 09 cd  ................
02b57e54  21 b8 01 4c cd 21 54 68-69 73 20 70 72 6f 67 72  !..L.!This progr
```

Case Study 2: .NET Unpacker

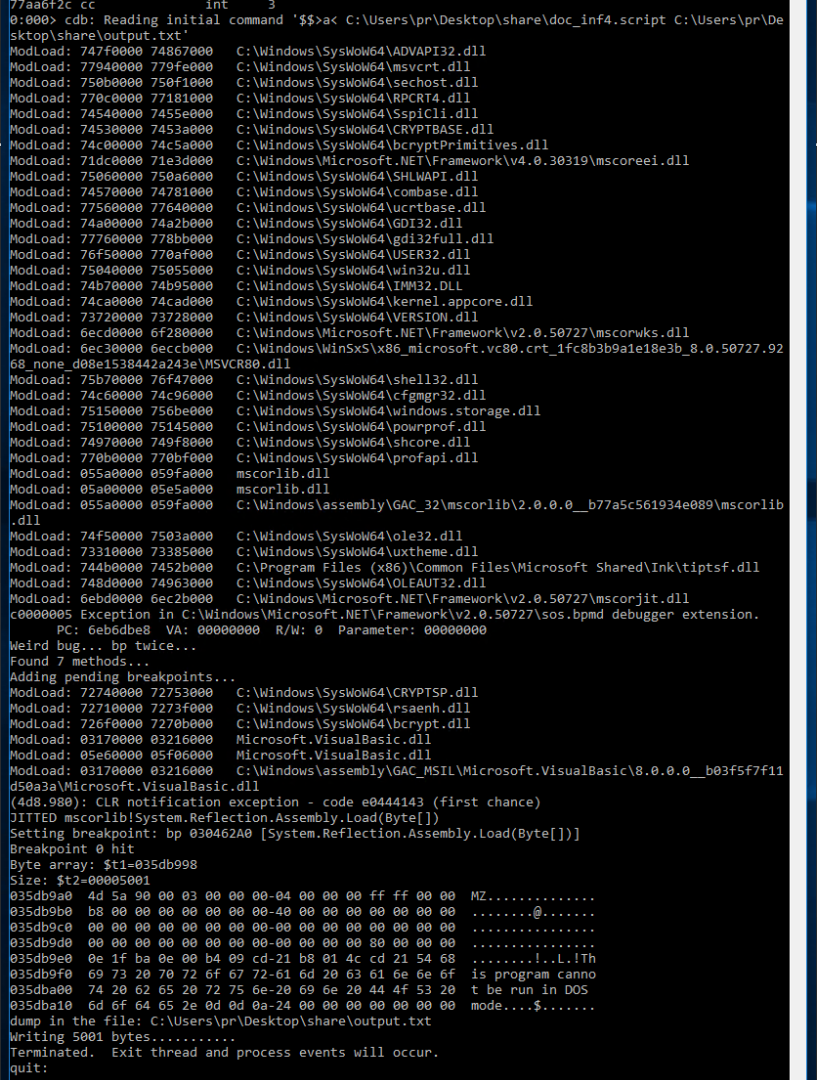
- Second step: WinDBG debugging
- Dump the unpacked PE

```
0:000> r ecx
ecx=02b57e04
0:000> .writemem c:\\output.exe ecx+0x8 L0x160c
Writing 5644 bytes……..
```

TALOS

Case Study 2:

- Massive analysis



```
77aa6f2c cc              int     3
0:000> cdb: Reading initial command '$$>a< C:\Users\pr\Desktop\share\doc_inf4.script C:\Users\pr\Desktop\share\output.txt'
ModLoad: 747f0000 74867000   C:\Windows\SysWoW64\ADVAPI32.dll
ModLoad: 77940000 779fe000   C:\Windows\SysWoW64\msvcrt.dll
ModLoad: 750b0000 750f1000   C:\Windows\SysWoW64\sechost.dll
ModLoad: 770c0000 77181000   C:\Windows\SysWoW64\RPCRT4.dll
ModLoad: 74540000 7455e000   C:\Windows\SysWoW64\SspiCli.dll
ModLoad: 74530000 7453a000   C:\Windows\SysWoW64\CRYPTBASE.dll
ModLoad: 74c00000 74c5a000   C:\Windows\SysWoW64\bcryptPrimitives.dll
ModLoad: 71dc0000 71e3d000   C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoreei.dll
ModLoad: 75060000 750a6000   C:\Windows\SysWoW64\SHLWAPI.dll
ModLoad: 74570000 74781000   C:\Windows\SysWoW64\combase.dll
ModLoad: 77560000 77640000   C:\Windows\SysWoW64\ucrtbase.dll
ModLoad: 74a00000 74a2b000   C:\Windows\SysWoW64\GDI32.dll
ModLoad: 77760000 778bb000   C:\Windows\SysWoW64\gdi32full.dll
ModLoad: 76f50000 770af000   C:\Windows\SysWoW64\USER32.dll
ModLoad: 75040000 75055000   C:\Windows\SysWoW64\win32u.dll
ModLoad: 74b70000 74b95000   C:\Windows\SysWoW64\IMM32.DLL
ModLoad: 74ca0000 74cad000   C:\Windows\SysWoW64\kernel.appcore.dll
ModLoad: 73720000 73728000   C:\Windows\SysWoW64\VERSION.dll
ModLoad: 6ecd0000 6f280000   C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorwks.dll
ModLoad: 6ec30000 6eccb000   C:\Windows\WinSxS\x86_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.9268_none_d08e1538442a243e\MSVCR80.dll
ModLoad: 75b70000 76f47000   C:\Windows\SysWoW64\shell32.dll
ModLoad: 74c60000 74c96000   C:\Windows\SysWoW64\cfgmgr32.dll
ModLoad: 75150000 756be000   C:\Windows\SysWoW64\windows.storage.dll
ModLoad: 75100000 75145000   C:\Windows\SysWoW64\powrprof.dll
ModLoad: 74970000 749f8000   C:\Windows\SysWoW64\shcore.dll
ModLoad: 770b0000 770bf000   C:\Windows\SysWoW64\profapi.dll
ModLoad: 055a0000 059fa000   mscorlib.dll
ModLoad: 05a00000 05e5a000   mscorlib.dll
ModLoad: 055a0000 059fa000   C:\Windows\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\mscorlib.dll
ModLoad: 74f50000 7503a000   C:\Windows\SysWoW64\ole32.dll
ModLoad: 73310000 73385000   C:\Windows\SysWoW64\uxtheme.dll
ModLoad: 744b0000 7452b000   C:\Program Files (x86)\Common Files\Microsoft Shared\Ink\tiptsf.dll
ModLoad: 748d0000 74963000   C:\Windows\SysWoW64\OLEAUT32.dll
ModLoad: 6ebd0000 6ec2b000   C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorjit.dll
c0000005 Exception in C:\Windows\Microsoft.NET\Framework\v2.0.50727\sos.bpmd debugger extension.
      PC: 6eb6dbe8  VA: 00000000  R/W: 0  Parameter: 00000000
Weird bug... bp twice...
Found 7 methods...
Adding pending breakpoints...
ModLoad: 72740000 72753000   C:\Windows\SysWoW64\CRYPTSP.dll
ModLoad: 72710000 7273f000   C:\Windows\SysWoW64\rsaenh.dll
ModLoad: 726f0000 7270b000   C:\Windows\SysWoW64\bcrypt.dll
ModLoad: 03170000 03216000   Microsoft.VisualBasic.dll
ModLoad: 05e60000 05f06000   Microsoft.VisualBasic.dll
ModLoad: 03170000 03216000   C:\Windows\assembly\GAC_MSIL\Microsoft.VisualBasic\8.0.0.0__b03f5f7f11d50a3a\Microsoft.VisualBasic.dll
(4d8.980): CLR notification exception - code e0444143 (first chance)
JITTED mscorlib!System.Reflection.Assembly.Load(Byte[])
Setting breakpoint: bp 030462A0 [System.Reflection.Assembly.Load(Byte[])]
Breakpoint 0 hit
Byte array: $t1=035db998
Size: $t2=00005001
035db9a0  4d 5a 90 00 03 00 00 00-04 00 00 00 ff ff 00 00  MZ..............
035db9b0  b8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  ........@.......
035db9c0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
035db9d0  00 00 00 00 00 00 00 00-00 00 00 00 80 00 00 00  ................
035db9e0  0e 1f ba 0e 00 b4 09 cd-21 b8 01 4c cd 21 54 68  ........!..L.!Th
035db9f0  69 73 20 70 72 6f 67 72-61 6d 20 63 61 6e 6e 6f  is program canno
035dba00  74 20 62 65 20 72 75 6e-20 69 6e 20 44 4f 53 20  t be run in DOS
035dba10  6d 6f 64 65 2e 0d 0d 0a-24 00 00 00 00 00 00 00  mode....$.......
dump in the file: C:\Users\pr\Desktop\share\output.txt
Writing 5001 bytes..........
Terminated.  Exit thread and process events will occur.
quit:
```

TALOS

PYKD Script

PYKD Script

- Python support for WinDBG: https://pykd.codeplex.com/
- Python script to automate analysis: https://github.com/Cisco-Talos/dotNET_WinDBG
- Features: dump relevant argument (string, integer…), dump byte arrays from Assembly.Load()…

TALOS

PYKD Script

```
0:020> .loadby sos clr
0:020> .load pykd
0:020> !py c:\Users\lucifer\DotNETPlugin.py
{
  "date": 1500306926,
  "bp": "System.Diagnostics.Process.Start(System.Diagnostics.ProcessStartInfo)",
  "arguments": {
    "0": {
      "fields": {
        "0": {
          "Type": "System.String",
          "Name": "fileName",
          "string": "C:\\WINDOWS\\system32\\calc.exe"
        },
        "1": {
          "Type": "System.String",
          "Name": "arguments",
          "string": ""
        },
        "2": {
          "Type": "System.String",
          "Name": "directory",
          "string": "C:\\Users\\lucifer"
        },
        "3": {
          "Type": "System.String",
          "Name": "verb",
          "string": ""
        },
```

TALOS

PYKD Script

DEMO

TALOS

Conclusion

www.talosintelligence.com
blog.talosintel.com
@talossecurity
@r00tbsd