

**Digital Forensics Lab**

**Cyber Security Department**

CYL-2002

Fall 2024

Lab #03

Instructor:

Ubaid Ullah

Fahad Waheed

# Document Analysis

Documents are a common way of sending or storing information like messages, reports, videos, or ideas. MS Office documents, Images, and audio files are some commonly used formats in our day-to-day lives. However, beyond what we see written in a word document or hear in an audio file, these documents can also contain hidden messages or malicious code that may execute when we open them. In this lab, we’ll explore some techniques to analyze and examine these documents.

# Microsoft Office Documents

There are two main file formats used by Microsoft Office documents:

* OLE (Object Linking and Embedding)
* OOXML (Office Open XML)

## OLE

OLE (Object Linking and Embedding) was the file format used in early versions of Microsoft Office between 1997 and 2003. It defined a “file within a file” structure which allowed other files to be embedded into a file. For example, an Excel spreadsheet could be embedded within a Microsoft Word document.

It supported file extensions like .rtf, .doc, .ppt, and .xls, among others.

## OOXML

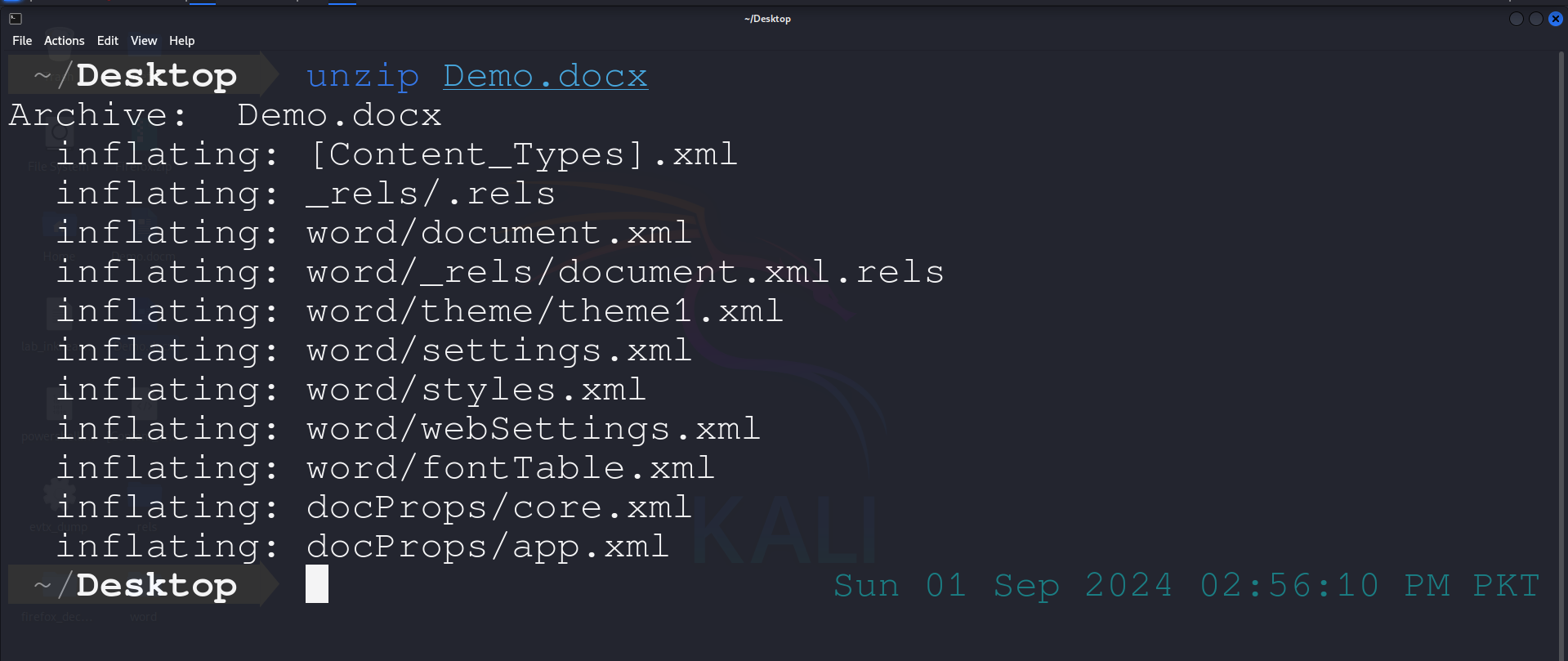
OOXML (Office Open XML) is the current file format used in Microsoft Office, which relies on an XML-based format for office documents.

The extensions for these documents include .docx, .pptx, and .xlsx, among others.

The OOXML format stores Office documents as ZIP containers. This means that the documents such as Word, Excel, and PowerPoint files, are actually just ZIP files. By renaming the extension from .docx, .xlsx, or .pptx to .zip, you can extract the contents of the archive and view the individual XML files. This is a useful feature for digital forensics, as it allows investigators to examine the contents of a document without modifying it.

# Anatomy of an OOXML document

To take an example, let’s create a word document with the text Hello World! in it, transfer it over to our Kali machine and then unzip it.



Here’s how the file structure of a Word document looks like:

.

├── [Content\_Types].xml

├── docProps

│   ├── app.xml

│   └── core.xml

├── \_rels

│   └── .rels

└── word

├── document.xml

├── fontTable.xml

├── \_rels

│   └── document.xml.rels

├── settings.xml

├── styles.xml

├── theme

│   └── theme1.xml

└── webSettings.xml

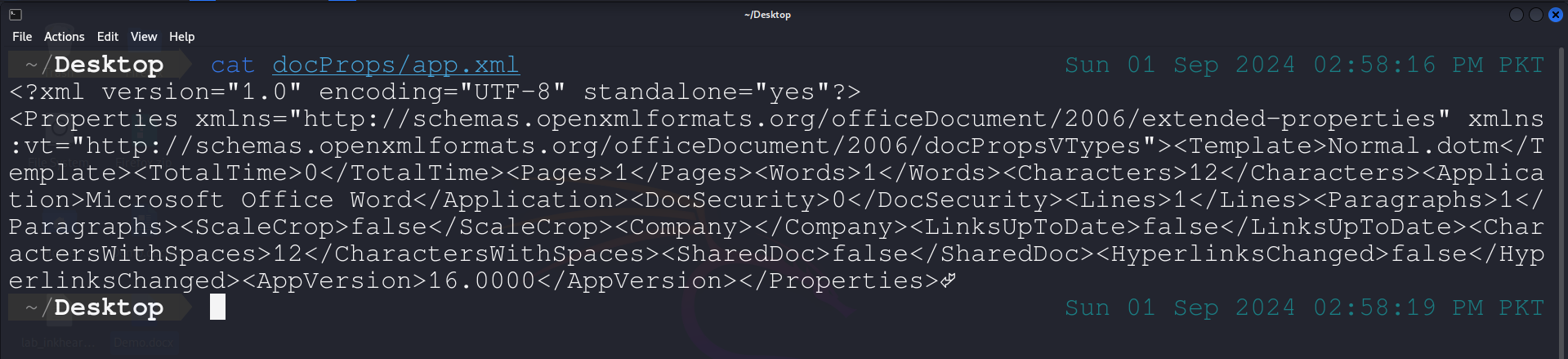
## [Content\_Types].xml

This file contains information about the content types that are present in the document and their corresponding file extensions.

## docProps

This folder contains two files, app.xml and core.xml.

app.xml — contains information about the application that was used to create the document.

 core.xml — contains metadata of the document, such as the author’s name, creation date, and modification date.

A screenshot of a computer

Description automatically generated

## \_rels

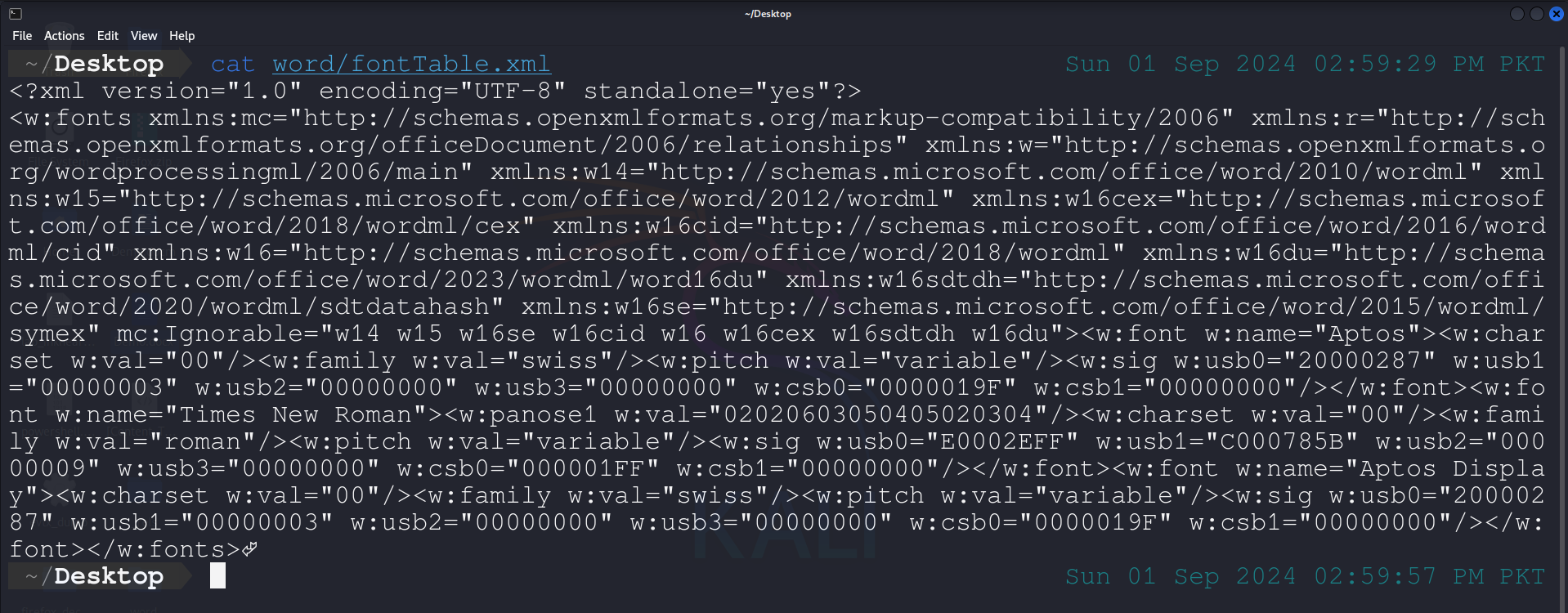
This folder contains one file named .rels.

* .rels — contains information about the relationships between the different parts of the document such as for app.xml and core.xml.

## word

This folder contains the actual content of the document.

document.xml — contains the actual text of the document.



* fontTable.xml — contains information about the fonts used in the document.
* **\_rels —** contains one file document.xml.rels.
* document.xml.rels — contains information about the relationships between the different parts of the document, such as styles, themes, settings, as well as the URIs for external links.
* settings.xml — contains document settings and configuration information.
* styles.xml — contains information about the styles used in the document.
* **theme** — contains files about the theme used in the document.
* theme1.xml — contains the actual theme content.
* webSettings.xml — contains information about the web-specific settings of the document, such as HTML frameset settings as well as how the document is handled when saved as HTML.

The information about any additional files that may be present in a document can be found on the link <http://officeopenxml.com/anatomyofOOXML.php>.

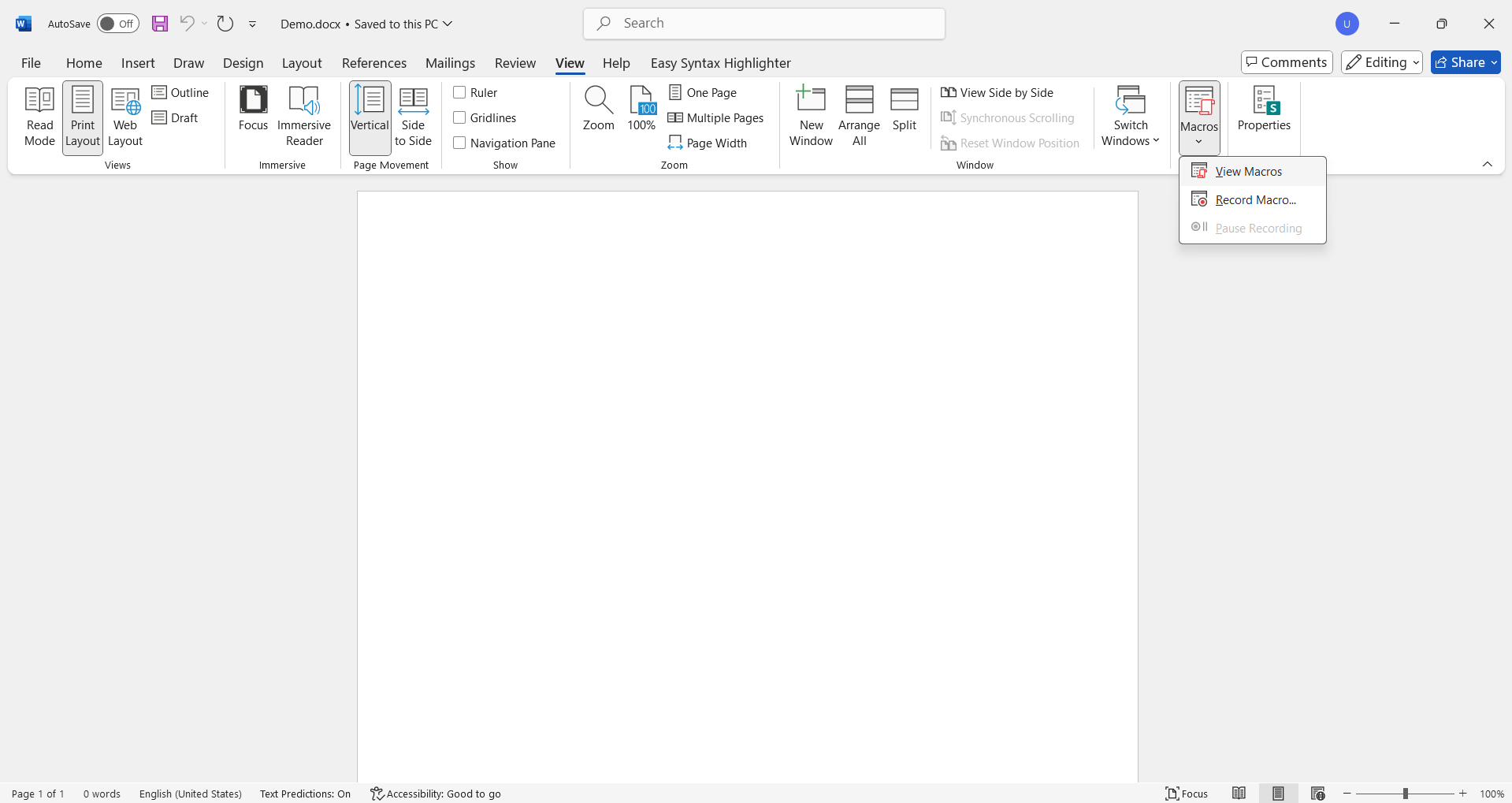
# Macro-Enabled Documents

Macro-Enabled documents are documents that contain macros, which are sets of instructions that automate tasks. Macros can be written in Visual Basic for Applications (VBA) and can be used to perform a wide range of tasks, such as formatting text, performing calculations, and automating complex processes. However, attackers often utilize this functionality of Office documents with a phishing attack and embed malicious macros to perform malicious actions and install malware on the system.

The extensions for these documents include .docm, .pptm, and .xlsm, among others.

To take an example, let’s create an empty word document, and follow the steps below:

1. Click View → Macros → View Macros.



1. Type a name such as HelloWorld, select Demo.docx (current document) under Macros in, and click create.

A screenshot of a computer

Description automatically generated

1. Paste the following code in the text box.

Sub HelloWorld()

Dim doc As Document

Set doc = Word.ActiveDocument

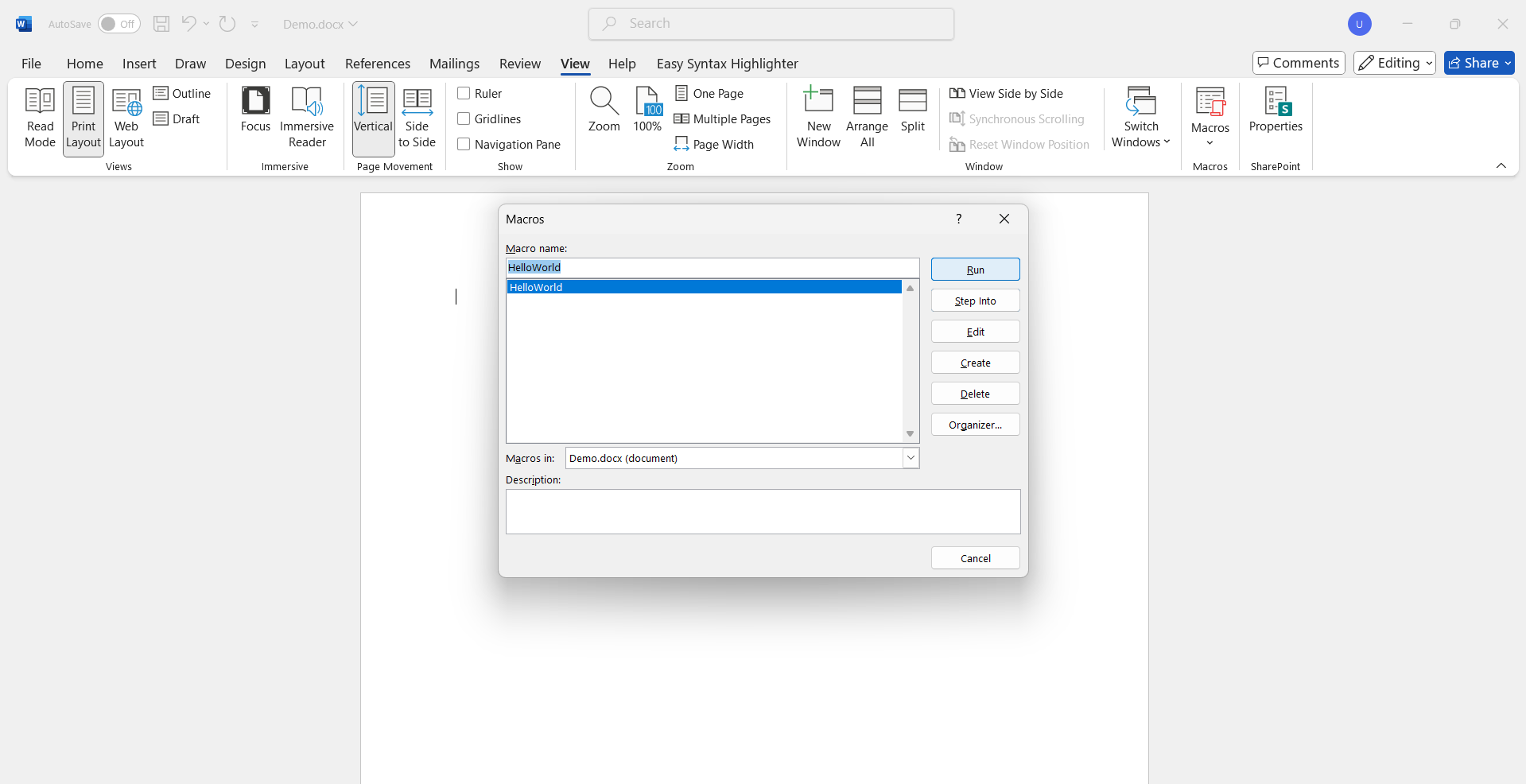
doc.Content.InsertAfter ("Hello, World!")

End Sub

A screenshot of a computer

Description automatically generated

1. Close the Microsoft Visual Basic for Application tab.
2. Repeat step 1, select the HelloWorld macro and click Run.

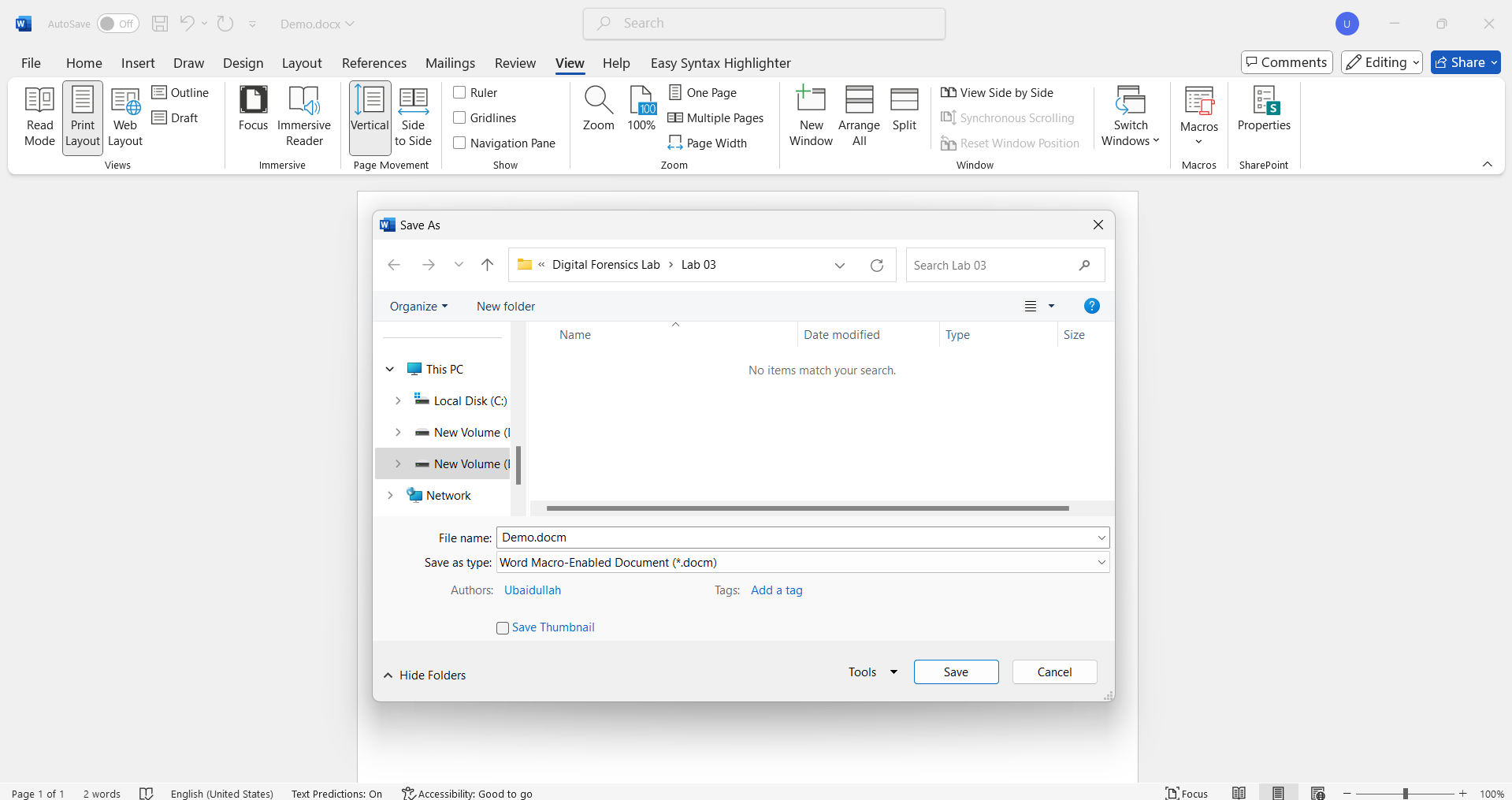


1. Observe that Hello, World! is now written in the document.

A screenshot of a computer

Description automatically generated

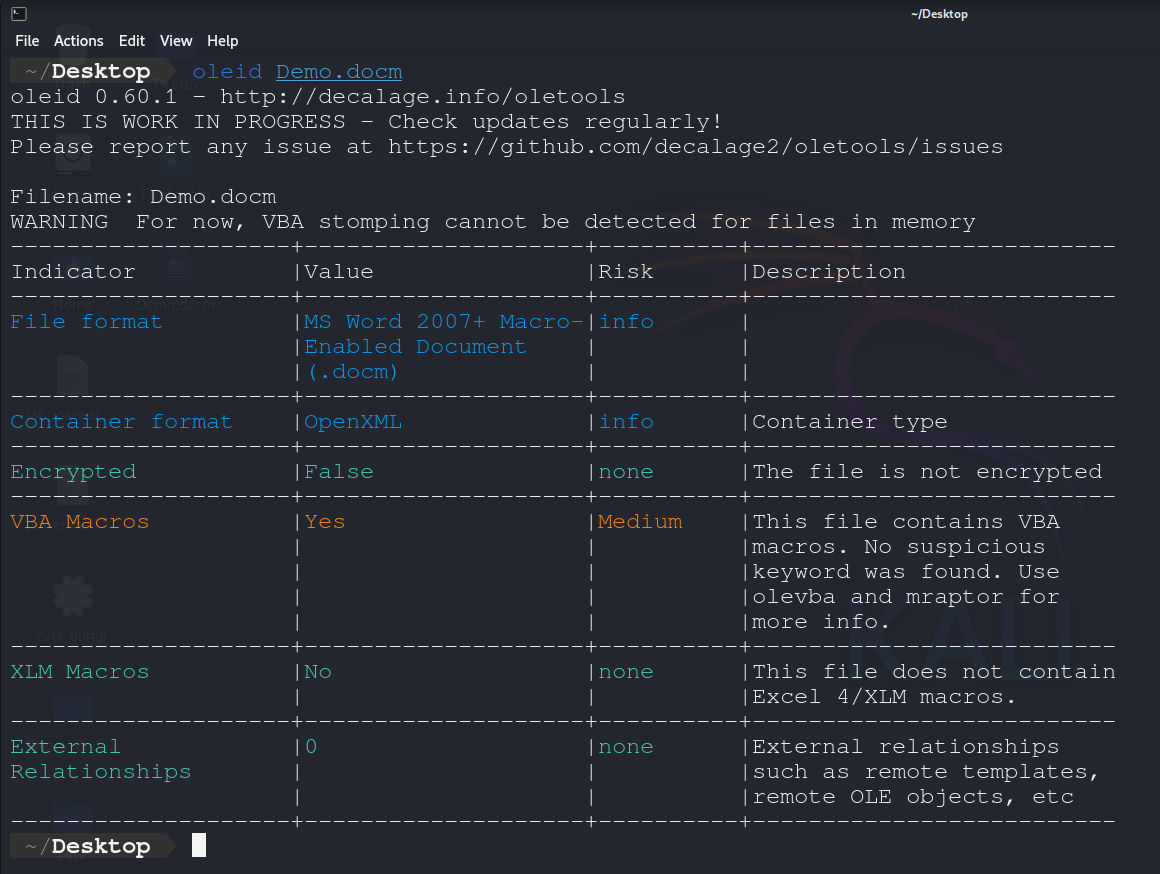
1. Save the document as Demo.docm.



As per the anatomy of OOXML files, the macro is now stored inside word/vbaProject.bin, however, we won’t be able to read it as it’s in binary form. But, we can use a collection of tools called oletools to analyze and extract macros from OLE files such as Microsoft Office Documents.

To install oletools, use the command: sudo -H pip install -U oletools.

Let’s use oleid to detect whether our document has any macros embedded in it.



The result shows that the tool found VBA Macros and evaluated the risk as medium. We can now proceed with using olevba to extract the macros from the document.

A screenshot of a computer

Description automatically generated

The output above shows that the extracted macro is exactly the same as we attached with the document.

Naming Convention: i22xxxx\_Lab03.pdf

The files provided for this task are potentially malicious. Please open and analyze them only in a sandbox or isolated environment.

DO NOT OPEN ANY MALICIOUS FILE ON WINDOWS HOST

Tool: sudo -H pip install -U oletools  
Task 01: Retrieve the two PDF documents from the “cw\_pdf\_files.7z” archive file. Perform a comprehensive analysis of the two files and present your findings, drawing conclusions as to whether each of the files may be a malicious PDF document. Use the following command to extract the archive:

7z x <filename>

Password: infected

Reference: <https://rohit12.medium.com/examining-a-pdf-file-using-two-tools-pdfid-and-pdf-parser-through-command-entered-into-a-661bcf99a11d>

<https://intezer.com/blog/incident-response/analyze-malicious-pdf-files/>

Task 02: Perform analysis of Word documents:

Password: infected

1. <https://github.com/HuskyHacks/PMAT-labs/raw/main/labs/3-1.GonePhishing-MaldocAnalysis/Word/docx/incrediblyPolishedResume.7z>
2. <https://github.com/HuskyHacks/PMAT-labs/raw/main/labs/3-1.GonePhishing-MaldocAnalysis/Excel/sheetsForFinancial.7z>

Note: To download the files directly from github into Kali use the command:

wget <link>