Lab Report

Name:	Inor Wang
Title:	Microsoft Office Document (OOXML Analysis)
Case:	25-T105
Date:	10/10/2025

Table of Contents

Document Revision History

Name	Revision Date	Version	Description
Inor Wang	10/10/2025	0.1	Draft

Executive Summary

On **October 10, 2025**, **Inor Wang** submitted a reproducible, offline examination of two Microsoft Word OOXML files, document.docx and targets.docx, to **Professor Jacob D. Stauffer**. The objective was to identify file type/protection, extract authorship and timeline metadata, map embedded and linked media, and recover encrypted content. Integrity was preserved with cryptographic hashing; all actions were documented with exact commands and annotated screenshots. Analysis was performed on macOS for triage/XML review and on Windows (NVIDIA RTX 3080, CUDA) for GPU-accelerated password recovery. Primary tools: **VS Code** (XML inspection), **Python 3/office2hashcat.py** (hash extraction), **Hashcat** (mode 9600), and **Microsoft Word** (visual confirmation).

Key findings from the examination are as follows:

- **Evidence integrity & file classification**: SHA-256 of document.docx is "32a073cf92796e9e6db13bc0afc5cb3550fe1005daa4fdc204cfdbf7933d249d" ; targets.docx is "a579629d929ffdbf89842e45971dc71d3444c172fabcbfeac48fb210d68724" ; document.docx = ZIP-based OOXML (50 4B 03 04) ; targets.docx = CDFV2 (encrypted) container (D0 CF 11 E0 …).
- **Protection status & recovery**: targets.docx confirmed password-protected (Office 2013) ; hash extracted via office2hashcat.py ($office$*2013*…) ; Hashcat command used: ./hashcat -a 0 -m 9600 -o cracked.txt hash.txt merged.txt ; Password successfully recovered: "protected." Decryption verified by opening the file.
- **Metadata (document.docx):** Creator: Guy Fawkes; Last Modified By: Guy Fawkes ; Created: 2024-02-20 14:38:00Z ; Modified: 2024-02-20 14:48:00Z (UTC) ; Revision info: <cp:revision> = 1 ; rsidRoot = 00787AF0.
- **Metadata (targets.docx):** Creator: Guy Faux ; Last Modified By: Inor Wang (expected due to password-removal/save for analysis) ; Created: 2024-02-20 15:03:00Z ; Last Modified: 2025-10-10 04:58:00Z (UTC).
- **Embedded/linked media (document.docx):** Two embedded images in document.xml (Picture 1 & Picture 2) with 10-digit IDs 1015826424 and 1199814006 ; descriptions match captions shown in screenshots ; One additional, unused media file present in /word/media (image3.png) not referenced by the document body (orphaned/leftover) ; One external image

reference via INCLUDEPICTURE (HTTPS URL), indicating a linked (not embedded) graphic

- **Embedded/linked media (targets.docx):** Three embedded images in targets.xml (Picture 1, 3, and 5) with resolve to image1.jpeg, image2.jpeg, and image.jpeg ; No additional images beyond the three in /word/media ; header/footer carry no images ; Page content labeled "FOR TRAINING PURPOSES ONLY" with three target photographs (scenic house/trees; park with statue; narrow urban street).

The OOXML artifacts fully address the lab's questions on file type, protection, authorship, edit timeline, revision markers, and media. Targets.docx was encrypted with Office 2013 and the password "protected" (recovered successfully), while document.docx was a standard OOXML ZIP. Authorship and UTC timestamps were confirmed from docProps/core.xml (creator: Guy Fawkes/Faux; last modified by: Guy Fawkes for document.docx and Inor Wang post-decryption for targets.docx). Revision indicators were documented. Media mapping showed two embedded images in document.docx plus one orphaned media file and an INCLUDEPICTURE web link, and three embedded images in targets.docx with no header/footer images. All findings derive from hashed, verifiable package parts and reproducible steps. The methods and results of this investigation should be considered in the context of learning how to navigate Microsoft OOXML to find actionable insights into investigations.

Synopsis

The examiner conducted an offline forensic review of Microsoft OOXML documents. After verifying evidence hashes, the examiner inspected each file's ZIP container and XML components to extract docProps metadata (creator, lastModifiedBy, created/modified), mapped relationships to embedded media, and compared images referenced in the document flow with those present in /word/media to identify unused remnants. For any protected documents, Office hashes were extracted and tested with Hashcat; results were recorded. The workflow is fully reproducible with explicit commands and annotated screenshots, and findings summarize provenance, edit timeline, protection status, and media/relationship discrepancies.

Client Questions:

1. What is the SHA256 hash of each file?
2. What is the type of each file? (e.g., Word, Excel, PowerPoint, etc.)
3. Are there any protected documents? Which files are protected?
4. document.docx
    a. Who is the creator of this document?
    b. Who last modified this document?
    c. When was this document created?
    d. When was this document last modified?
    e. How many revision identifiers (rsid) are associated to this document?
    f. What is the rsidRoot of this document?
    g. How many images are in the document.xml file? (Must open the document)
    h. What is the 10-digit ID and the description (descr) of each image?
    i. How many images are really in this document?
    j. If additional images are found, what are the filenames and describe the contents.
5. targets.docx
    a. What is the output from office2hashchat?
    b. What version of Office does this document use?
    c. What is the full hashcat command you used to crack this password hash?
    d. What was the cracked password?

e. Open the document using the password. Describe the contents.

f. In Word, go to "Protect Document" and remove the password, then save the file. Proceed with your regular analysis of this unprotected file.

    i. Who is the creator of this document?

    ii. Who last modified this document?

    iii. When was this document created?

    iv. When was this document last modified?

    v. How many images are in this document?

    vi. If additional images are found, what are the filenames and describe the contents.

Scope of Work:

- Acquisition of the evidence file from Professor Stauffer in the UTSA Canvas website.
- Analysis of Microsoft OOXML documents using VSCode, Python3, Office2Hashcat.py, Microsoft Word and Hashcat.
- Verification of evidentiary integrity using MD5, SHA1, and SHA256 cryptographic hashes.

Evidence Analyzed

This section provides details of the digital evidence collected

Evidence ID	**E001**
Name	ooxml_documents.zip
Type	Zip archive data, at least v2.0 to extract, compression method=deflate
Size	2.7 MB
MD5	a0f31b814ffef0021971c2bbbcf86761
SHA1	47a55fa91de7ac23907ad78771d54ad64a53a348
SHA256	69f9696a8717e2910c21da91f259ba1be6ce4b19579d50bddcaf96c92af51bb3

Tools Used

Workstation

Hostname	Operating System	Build	Physical / Virtual	Built
Inor	Windows 11	2023	Physical	2/15/2015
Inors-Macbook-Air	MacOS Sequoia	2024	Physical	12/03/2020

Software

Name	Version	Release	Purpose
VSCode (Microsoft)	1.94.6 (Stable)	Oct 2024	Code editor used to open OOXML packages as ZIPs and review XML parts (docProps/core.xml, word/document.xml, word/_rels/document.xml.rels) with syntax highlighting and quick search to map metadata, relationships, and embedded media.
Python3 (Python)	3.12.3	2024	Runtime for helper scripts, including running office2hashcat.py, simple parsing of XML when needed, and any automation for listing ZIP members/hashes.
Office2Hashcat.py	N/A	N/A	Extracts the password hash from protected OOXML documents and outputs it in a Hashcat-ready format (correct mode for Office version).
Hashcat (Hashcat)	6.2.6	Jun 2021	Performs password-cracking attempts against the extracted Office hash (e.g., mode 9600/9500/9400 depending on Office type), recording success/failure, time, and attack parameters.
Word (Microsoft)	16.90	Oct 2024	Used to visually verify and access the decrypted contents of `targets.docx` after password recovery, confirm embedded media placement, inspect metadata through the "Info" panel, and validate document structure and labeling for comparison with extracted XML data.

Analysis Findings

Overview of Examination Procedures

The forensic analysis of the **two OOXML documents (document.docx, targets.docx)** was conducted on the examiner's macOS host with selective Windows execution for GPU-accelerated cracking. Evidence integrity was verified with SHA-256 hashing prior to analysis, and a step-by-step, reproducible workflow (explicit commands, paths, and annotated screenshots) was followed in accordance with the assignment rubric and template.

Additional targeted analysis was performed using:

- **Terminal (macOS) / PowerShell (Windows):** hashing, triage (file, xxd), unzip, and command logging
- **Visual Studio Code:** XML viewing/formatting and targeted searches across document.xml, core.xml, document.xml.rels, and settings.xml.
- **office2hashcat.py:** extraction of Hashcat-ready Office hash from targets.docx.
- **Hashcat (GPU):** password recovery (mode 9600) and verification of cracked credentials.
- **Microsoft Word:** visual verification of embedded images and final content confirmation post-decryption.

Throughout the process, all findings were documented, and cryptographic hash values were maintained for validation.

Evidence Reviewed

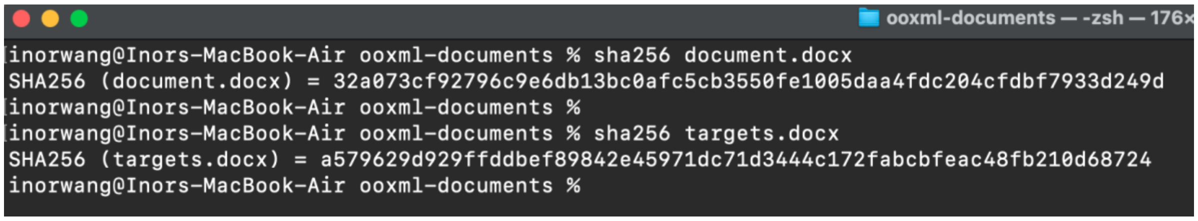
1. **ooxml_documents.zip (E001)**: Zip file containing OOXML documents

Key Findings

1. What is the SHA256 hash of each file?

- **Analysis Performed:**
    - The document.docx and target.docx files were examined through the examiner's macOS host computer's terminal to get the sha256 hash as shown in Figure 1.
    - Commands: "*sha256 document.docx*" and "*sha256 targets.docx*"
- **Answer:**
  The SHA256 hash of "document.docx" is: "**32a073cf92796e9e6db13bc0afc5cb3550fe1005daa4fdc204cfdbf7933d249d**". The SHA256 hash of "targets.docx" is: "**a579629d929ffdbf89842e45971dc71d3444c172fabcbfeac48fb210d68724**".

- **Supporting Evidence:**



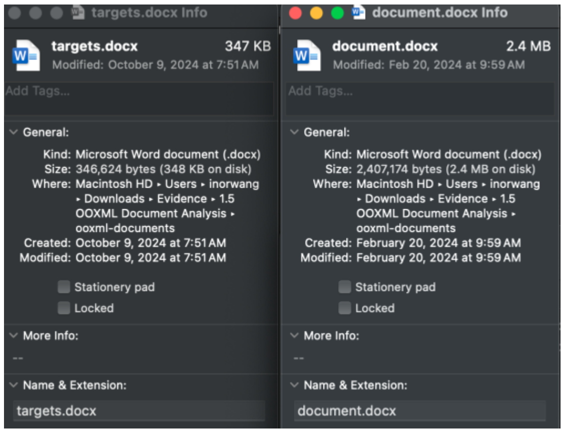
```
inorwang@Inors-MacBook-Air ooxml-documents % sha256 document.docx
SHA256 (document.docx) = 32a073cf92796c9e6db13bc0afc5cb3550fe1005daa4fdc204cfdbf7933d249d
inorwang@Inors-MacBook-Air ooxml-documents %
inorwang@Inors-MacBook-Air ooxml-documents % sha256 targets.docx
SHA256 (targets.docx) = a579629d929ffddbef89842e45971dc71d3444c172fabcbfeac48fb210d68724
inorwang@Inors-MacBook-Air ooxml-documents %
```

Figure 1. Examiner utilizing the sha256 command to obtain the sha256 hashes of document.docx and targets.docx

2. What is the type of each file? (e.g., Word, Excel, PowerPoint, etc.)

- **Analysis Performed:**
    - The document.docx and target.docx files were examined through the examiner's macOS host computer.
    - As shown in Figure 2, the extension of both the document.docx and target.docx files are ".docx" however after further search within the macOS terminal using the "file" command, the document.docx file is a Microsoft OOXML file and the targets.docx file is a CDFV2 Encrypted file as shown in Figure 3.
    - Additionally as shown in Figure 4, the first 4 bytes is "PK .." which indicates that is is compressed.
    - Commands: "*file document.docx*" and "*file targets.docx*"
- **Answer:**
  document.docx is a Microsoft OOXML Word Document (.docx) and targets.docx is a Microsoft OOXML Word Document (.docx) as well, however it is CDFV2 Encrypted with a password. CDFV2 is Microsoft Word's older encryption layer.

- **Supporting Evidence:**



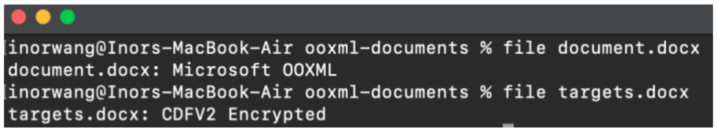
Modified: October 9, 2024 at 7:51 AM

General:
Kind: Microsoft Word document (.docx)
Size: 346,624 bytes (348 KB on disk)
Where: Macintosh HD ▸ Users ▸ inorwang ▸ Downloads ▸ Evidence ▸ 1.5 OOXML Document Analysis ▸ ooxml-documents
Created: October 9, 2024 at 7:51 AM
Modified: October 9, 2024 at 7:51 AM
Stationery pad
Locked
More Info:
--
Name & Extension:
targets.docx

Modified: Feb 20, 2024 at 9:59 AM

General:
Kind: Microsoft Word document (.docx)
Size: 2,407,174 bytes (2.4 MB on disk)
Where: Macintosh HD ▸ Users ▸ inorwang ▸ Downloads ▸ Evidence ▸ 1.5 OOXML Document Analysis ▸ ooxml-documents
Created: February 20, 2024 at 9:59 AM
Modified: February 20, 2024 at 9:59 AM
Stationery pad
Locked
More Info:
--
Name & Extension:
document.docx

Figure 2.Screenshot of the information tab for targets.docx and document.docx



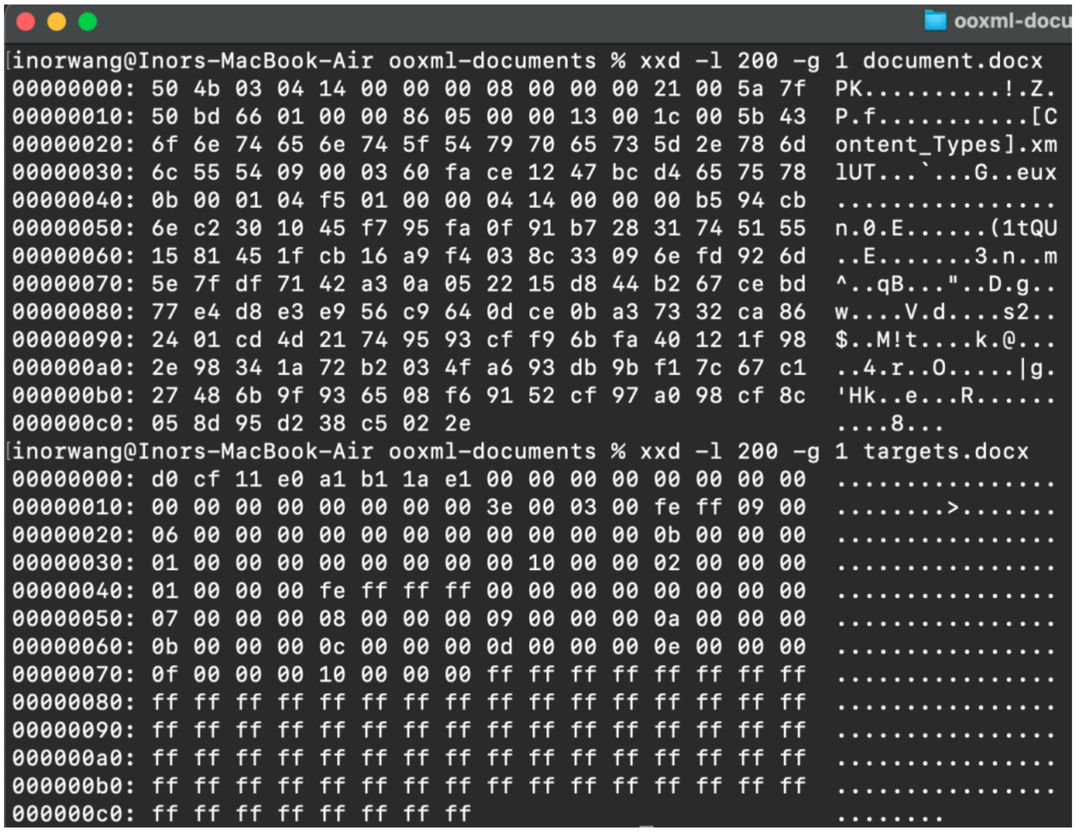
```
inorwang@Inors-MacBook-Air ooxml-documents % file document.docx
document.docx: Microsoft OOXML
inorwang@Inors-MacBook-Air ooxml-documents % file targets.docx
targets.docx: CDFV2 Encrypted
```

Figure 3. Screenshot of the output of the "file" command for document.docx and targets.docx

3. Are there any protected documents? Which files are protected?

- **Analysis Performed:**
  - The document.docx and target.docx files were examined through the examiner's macOS host computer's terminal.
  - The hex view of document.docx begins with "50 4B 03 04", the signature for zip-based OOXML files, confirming it is a normal, unencrypted/protected Word document as shown in Figure 4.
  - The hex view of targets.docx begins with "D0 CF 11 E0 A1 B1 1A E1", which indicates a Compound File Binary Format (CDFV2), used by encrypted or legacy Word documents, showing that targets.docx is protected with a password as shown in Figure 4.
  - As shown in Figure 3, using the "file" command against the two files, the targets.docx file is CDFV2 encrypted which further corroborates that the file is protected.
  - Commands: "*xxd -l 200 -g 1 document.docx*" and "*xxd -l 200 -g 1 targets.docx*"
- **Answer:**
  As shown in Figures 3 and 4, the **target.docx file is password-protected while the document.docx file is not.**

- **Supporting Evidence:**



```
inorwang@Inors-MacBook-Air ooxml-documents % xxd -l 200 -g 1 document.docx
00000000: 50 4b 03 04 14 00 00 08 00 00 00 21 00 5a 7f  PK.........!.Z.
00000010: 50 bd 66 01 00 00 86 05 00 00 13 00 1c 00 5b 43  P.f...........[C
00000020: 6f 6e 74 65 6e 74 5f 54 79 70 65 73 5d 2e 78 6d  ontent_Types].xm
00000030: 6c 55 54 09 00 03 60 fa ce 12 47 bc d4 65 75 78  lUT...`...G..eux
00000040: 0b 00 01 04 f5 01 00 00 04 14 00 00 00 b5 94 cb  ................
00000050: 6e c2 30 10 45 f7 95 fa 0f 91 b7 28 31 74 51 55  n.0.E......(1tQU
00000060: 15 81 45 1f cb 16 a9 f4 03 8c 33 09 6e fd 92 6d  ..E.......3.n..m
00000070: 5e 7f df 71 42 a3 0a 05 22 15 d8 44 b2 67 ce bd  ^..qB..."..D.g..
00000080: 77 e4 d8 e3 e9 56 c9 64 0d ce 0b a3 73 32 ca 86  w....V.d....s2..
00000090: 24 01 cd 4d 21 74 95 93 cf f9 6b fa 40 12 1f 98  $..M!t....k.@...
000000a0: 2e 98 34 1a 72 b2 03 4f a6 93 db 9b f1 7c 67 c1  ..4.r..O.....|g.
000000b0: 27 48 6b 9f 93 65 08 f6 91 52 cf 97 a0 98 cf 8c  'Hk..e...R......
000000c0: 05 8d 95 d2 38 c5 02 2e                          ....8...
inorwang@Inors-MacBook-Air ooxml-documents % xxd -l 200 -g 1 targets.docx
00000000: d0 cf 11 e0 a1 b1 1a e1 00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00 3e 00 03 00 fe ff 09 00  ........>.......
00000020: 06 00 00 00 00 00 00 00 00 00 00 00 0b 00 00 00  ................
00000030: 01 00 00 00 00 00 00 00 00 10 00 00 02 00 00 00  ................
00000040: 01 00 00 00 fe ff ff ff 00 00 00 00 00 00 00 00  ................
00000050: 07 00 00 00 08 00 00 00 09 00 00 00 0a 00 00 00  ................
00000060: 0b 00 00 00 0c 00 00 00 0d 00 00 00 0e 00 00 00  ................
00000070: 0f 00 00 00 10 00 00 00 ff ff ff ff ff ff ff ff  ................
00000080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
00000090: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
000000a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
000000b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
000000c0: ff ff ff ff ff ff ff ff                          ........
```

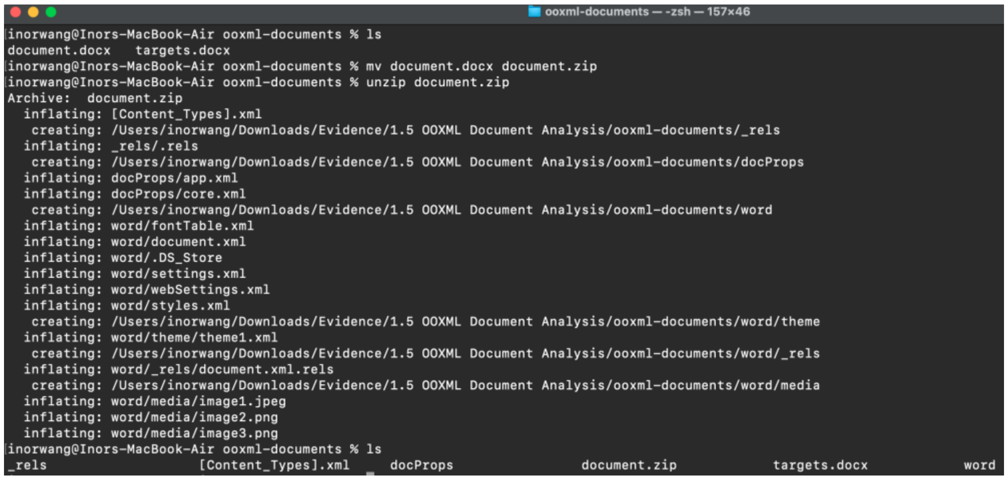
Figure 4. Screenshot of the hex output for document.docx and targets.docx, showing document.docx as a standard OOXML file and targets.docx as an encrypted CDFV2 file.

document.docx

1. Who is the creator of this document?

- **Analysis Performed:**
    - The document.docx document file was examined through the examiner's macOS host computer's terminal and Visual Studio Code application.
    - The .docx file was renamed to .zip and extracted using the unzip command, revealing its OOXML (Open Office XML) structure as shown in Figure 5.
        - Commands: "*mv document.docx document.zip*" and "*unzip document.zip*"
    - The examiner installed XML Tools within Visual Studio Code to easily analyze and format the extracted XML contents from the .docx file as shown in Figure 6.
    - The core.xml file contains the document's metadata, including details such as the creator, title, revision number, and creation/modification timestamps.
    - The examiner formatted the core.xml file to easily analyze the contents, which contained who the creator of the document is ("Guy Fawkes"), as shown in Figure 7.
- **Answer:**
  The creator of the document.docx file is "**Guy Fawkes**", as shown in Figure 7.

- **Supporting Evidence:**



```
ooxml-documents — -zsh — 157×46
inorwang@Inors-MacBook-Air ooxml-documents % ls
document.docx   targets.docx
inorwang@Inors-MacBook-Air ooxml-documents % mv document.docx document.zip
inorwang@Inors-MacBook-Air ooxml-documents % unzip document.zip
Archive:  document.zip
  inflating: [Content_Types].xml
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/_rels
  inflating: _rels/.rels
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/docProps
  inflating: docProps/app.xml
  inflating: docProps/core.xml
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/word
  inflating: word/fontTable.xml
  inflating: word/document.xml
  inflating: word/.DS_Store
  inflating: word/settings.xml
  inflating: word/webSettings.xml
  inflating: word/styles.xml
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/word/theme
  inflating: word/theme/theme1.xml
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/word/_rels
  inflating: word/_rels/document.xml.rels
   creating: /Users/inorwang/Downloads/Evidence/1.5 OOXML Document Analysis/ooxml-documents/word/media
  inflating: word/media/image1.jpeg
  inflating: word/media/image2.png
  inflating: word/media/image3.png
inorwang@Inors-MacBook-Air ooxml-documents % ls
_rels                   [Content_Types].xml     docProps                document.zip            targets.docx            word
```

Figure 5. Screenshot showing document.docx renamed and unzipped, revealing its internal OOXML folder structure and XML components.

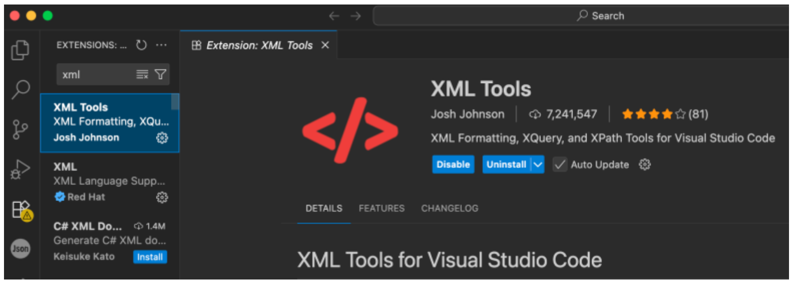
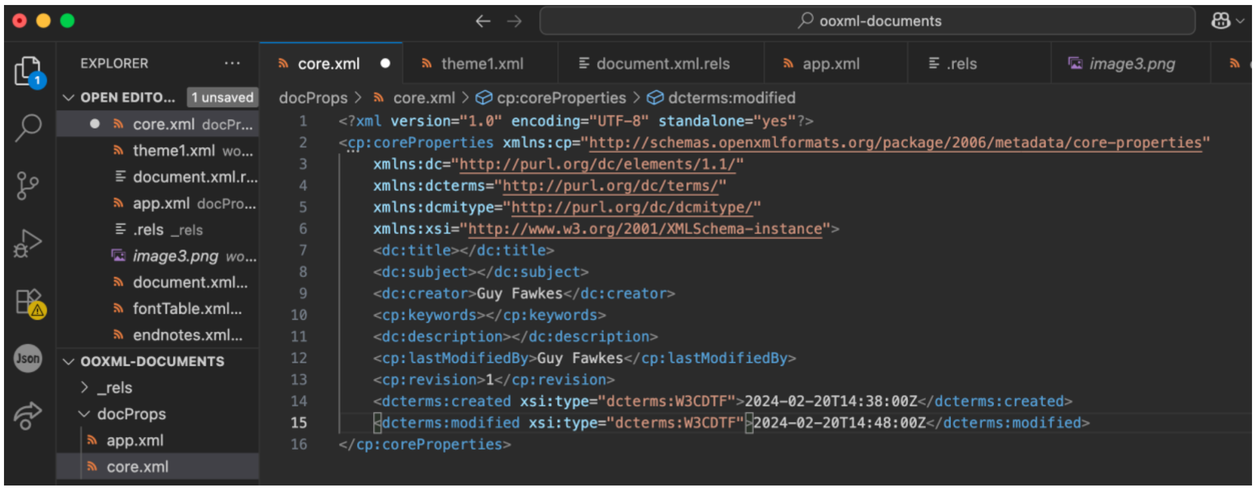


Figure 6. Screenshot of the XML Tools extension by Josh Johnson in Visual Studio Code, used for formatting, viewing, and analyzing XML files extracted from the .docx package.



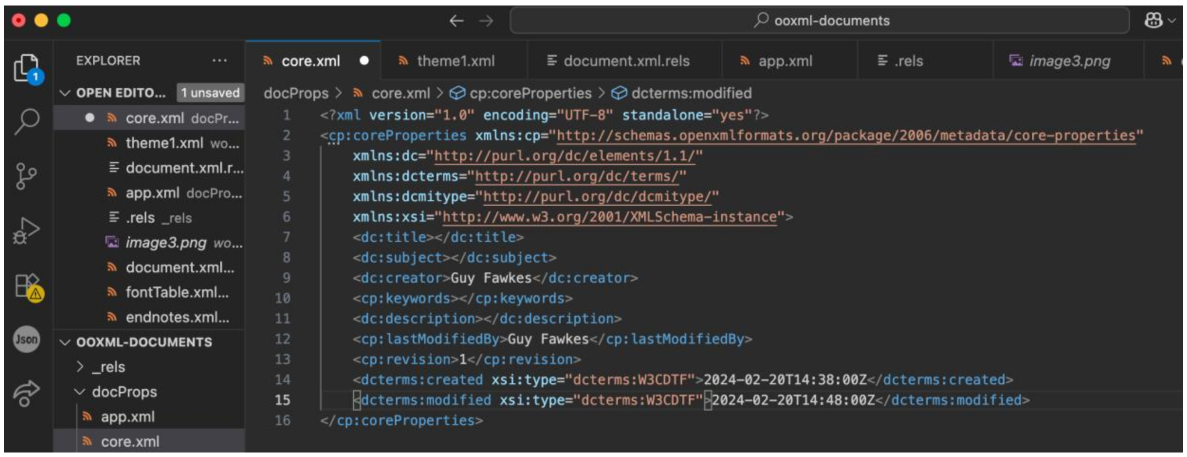
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Fawkes</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description></dc:description>
    <cp:lastModifiedBy>Guy Fawkes</cp:lastModifiedBy>
    <cp:revision>1</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T14:38:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2024-02-20T14:48:00Z</dcterms:modified>
</cp:coreProperties>
```

Figure 7. Screenshot of core.xml opened in Visual Studio Code showing document metadata for document.docx

2. Who last modified this document?

- **Analysis Performed:**
    - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
    - The core.xml file contains the document's metadata, including details such as the creator, title, revision number, and creation/modification timestamps.
    - The examiner formatted the core.xml file to easily analyze the contents, which contained who last modified the document ("Guy Fawkes"), as shown in Figure 8.
- **Answer:**
  The person who last modified the document.docx file was "**Guy Fawkes**", as shown in Figure 8.

- **Supporting Evidence:**



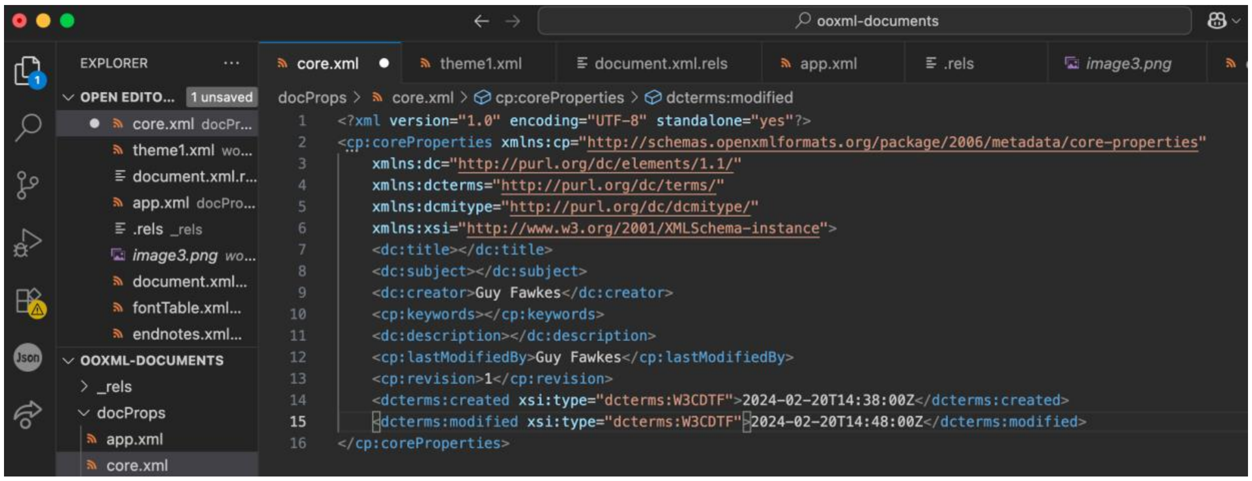
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Fawkes</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description></dc:description>
    <cp:lastModifiedBy>Guy Fawkes</cp:lastModifiedBy>
    <cp:revision>1</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T14:38:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2024-02-20T14:48:00Z</dcterms:modified>
</cp:coreProperties>
```

Figure 8. Screenshot of core.xml opened in Visual Studio Code showing document metadata for document.docx

3. When was this document created?

- **Analysis Performed:**
  - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
  - The core.xml file contains the document's metadata, including details such as the creator, title, revision number, and creation/modification timestamps.
  - The examiner formatted the core.xml file to easily analyze the contents, which contained when the document.docx was created ("2024-02-20T14:38:00Z"), as shown in Figure 9.
  - The timestamp ends with "Z", indicating the time is recorded in Coordinated Universal Time (UTC).
- **Answer:**
  The document.docx file was created on **February 20, 2024, at 14:38:00 UTC**, as shown in Figure 9.

- **Supporting Evidence:**



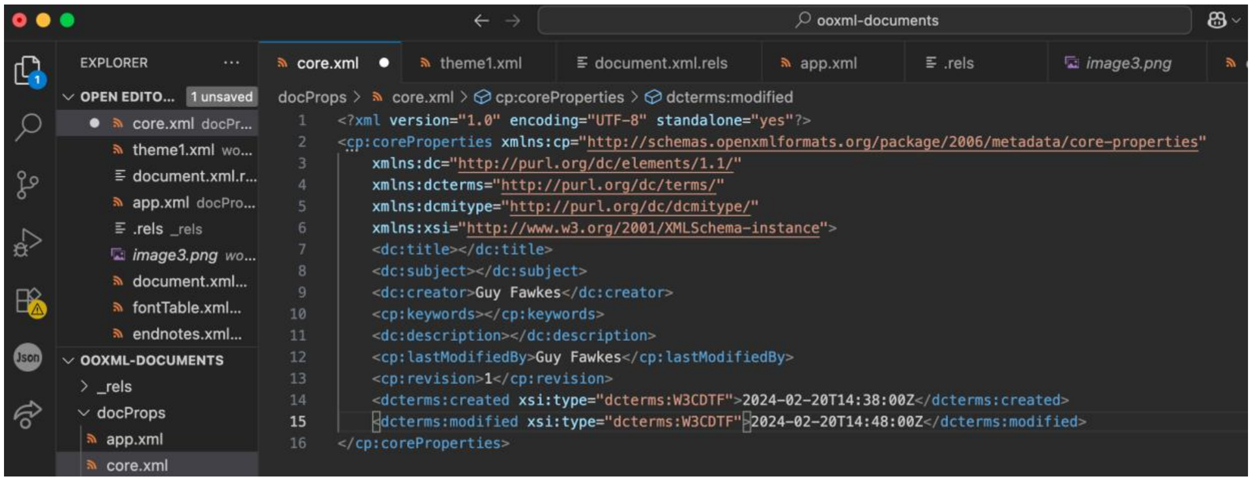
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Fawkes</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description></dc:description>
    <cp:lastModifiedBy>Guy Fawkes</cp:lastModifiedBy>
    <cp:revision>1</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T14:38:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2024-02-20T14:48:00Z</dcterms:modified>
</cp:coreProperties>
```

Figure 9. Screenshot of core.xml opened in Visual Studio Code showing document metadata for document.docx

4. When was this document last modified?

- **Analysis Performed:**
  - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
  - The core.xml file contains the document's metadata, including details such as the creator, title, revision number, and creation/modification timestamps.
  - The examiner formatted the core.xml file to easily analyze the contents, which contained when the document.docx was last modified ("2024-02-20T14:48:00Z"), as shown in Figure 10.
  - The timestamp ends with "Z", indicating the time is recorded in Coordinated Universal Time (UTC).
- **Answer:**
  The document.docx file was last modified on **February 20, 2024, at 14:48:00 UTC**, as shown in Figure 10.

- **Supporting Evidence:**



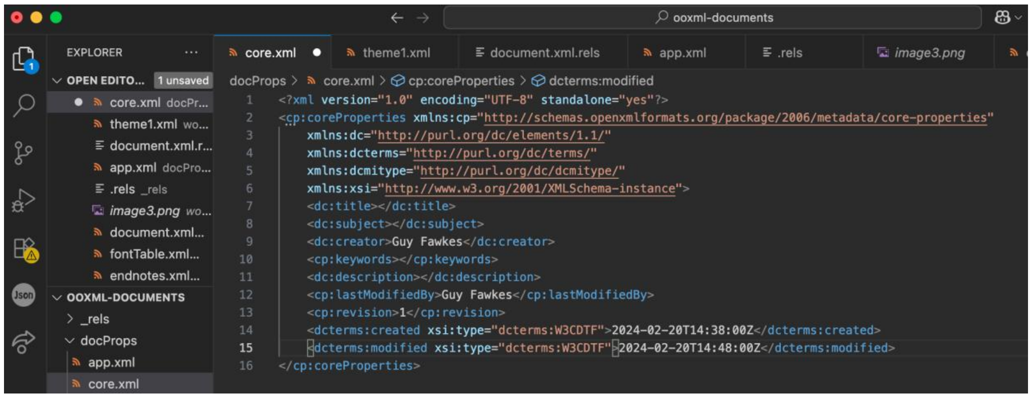
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Fawkes</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description></dc:description>
    <cp:lastModifiedBy>Guy Fawkes</cp:lastModifiedBy>
    <cp:revision>1</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T14:38:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2024-02-20T14:48:00Z</dcterms:modified>
</cp:coreProperties>
```

Figure 10. Screenshot of core.xml opened in Visual Studio Code showing document metadata for document.docx

5. How many revision identifiers (rsid) are associated to this document?

- **Analysis Performed:**
  - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
  - The core.xml file contains the document's metadata, including details such as the creator, title, revision number, and creation/modification timestamps.
  - The examiner formatted the core.xml file to easily analyze the contents, which contained how many revision identifiers are associated to the document ("1"), as shown in Figure 11.
- **Answer:**
  According to the core.xml file, the document contains **one revision identifier (rsid),** shown by the tag <cp:revision>1</cp:revision in Figure 11.

- **Supporting Evidence:**



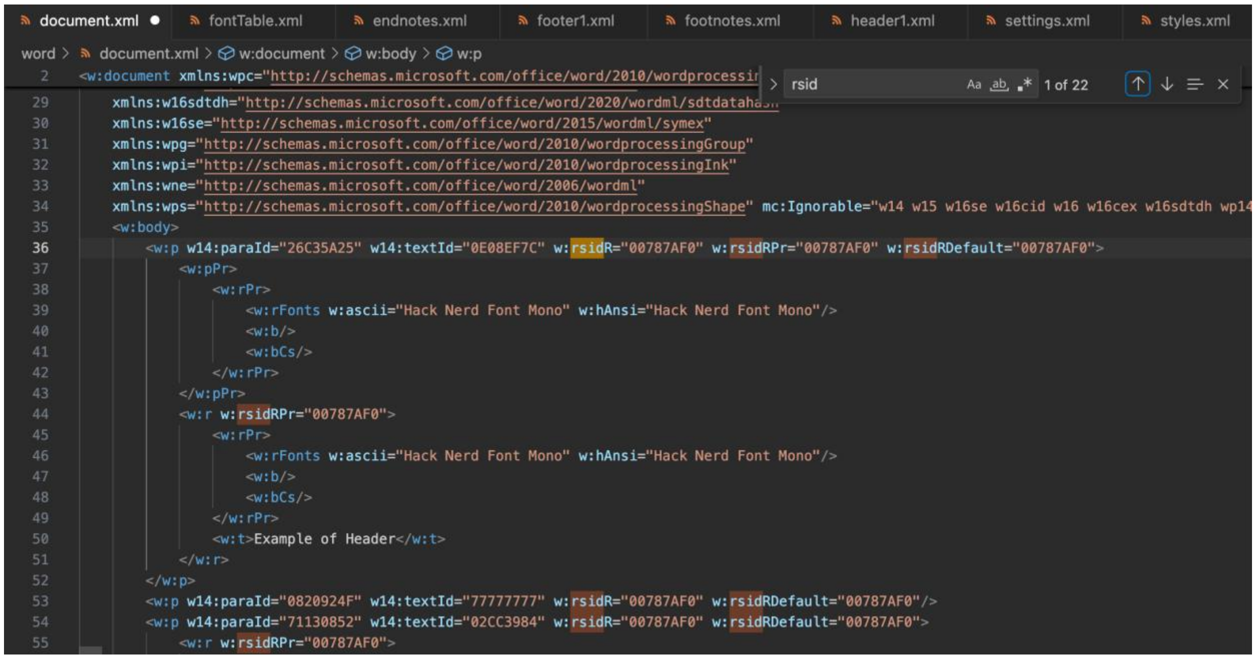
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Fawkes</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description></dc:description>
    <cp:lastModifiedBy>Guy Fawkes</cp:lastModifiedBy>
    <cp:revision>1</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T14:38:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2024-02-20T14:48:00Z</dcterms:modified>
</cp:coreProperties>
```

Figure 11. Screenshot of core.xml opened in Visual Studio Code showing document metadata for document.docx

6. What is the rsidRoot of this document?

- **Analysis Performed:**
    - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
    - The examiner searched the document.xml file for all instances of rsid to identify revision tracking information.
    - The examiner located the rsidRoot attribute (00787AF0) within the <w:document> tag, confirming the document's original revision identifier, as shown in Figure 12.
- **Answer:**
  The rsidRoot of this document is **00787AF0**, found in the <w:document> tag shown in Figure 12. The rsid values track different edits and revisions within the Word document, helping identify changes made across version.

- **Supporting Evidence:**



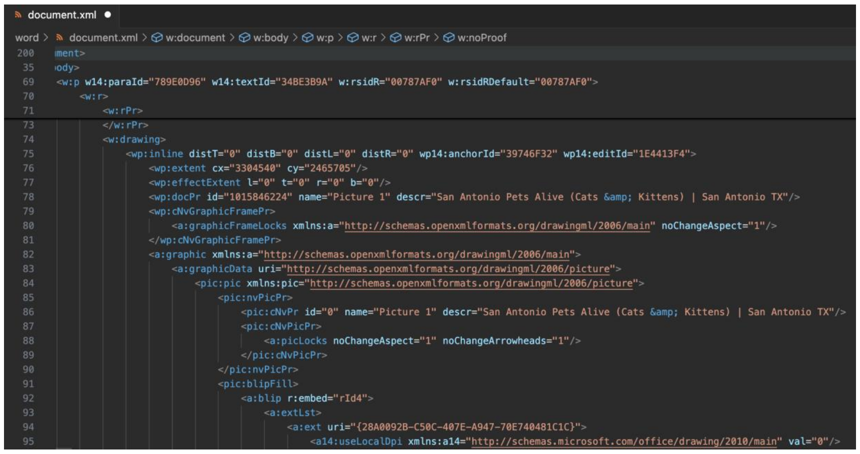
```
<w:document xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessi
    xmlns:w16sdtdh="http://schemas.microsoft.com/office/word/2020/wordml/sdtdataha
    xmlns:w16se="http://schemas.microsoft.com/office/word/2015/wordml/symex"
    xmlns:wpg="http://schemas.microsoft.com/office/word/2010/wordprocessingGroup"
    xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordprocessingInk"
    xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml"
    xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordprocessingShape" mc:Ignorable="w14 w15 w16se w16cid w16 w16cex w16sdtdh wp14
    <w:body>
        <w:p w14:paraId="26C35A25" w14:textId="0E08EF7C" w:rsidR="00787AF0" w:rsidRPr="00787AF0" w:rsidRDefault="00787AF0">
            <w:pPr>
                <w:rPr>
                    <w:rFonts w:ascii="Hack Nerd Font Mono" w:hAnsi="Hack Nerd Font Mono"/>
                    <w:b/>
                    <w:bCs/>
                </w:rPr>
            </w:pPr>
            <w:r w:rsidRPr="00787AF0">
                <w:rPr>
                    <w:rFonts w:ascii="Hack Nerd Font Mono" w:hAnsi="Hack Nerd Font Mono"/>
                    <w:b/>
                    <w:bCs/>
                </w:rPr>
                <w:t>Example of Header</w:t>
            </w:r>
        </w:p>
        <w:p w14:paraId="0820924F" w14:textId="77777777" w:rsidR="00787AF0" w:rsidRDefault="00787AF0"/>
        <w:p w14:paraId="71130852" w14:textId="02CC3984" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
            <w:r w:rsidRPr="00787AF0">
```

Figure 12. Screenshot of document.xml opened in Visual Studio Code showing multiple revision identifiers (rsid) and the document's rsidRoot value.

7. How many images are in the document.xml file? (Must open the document)

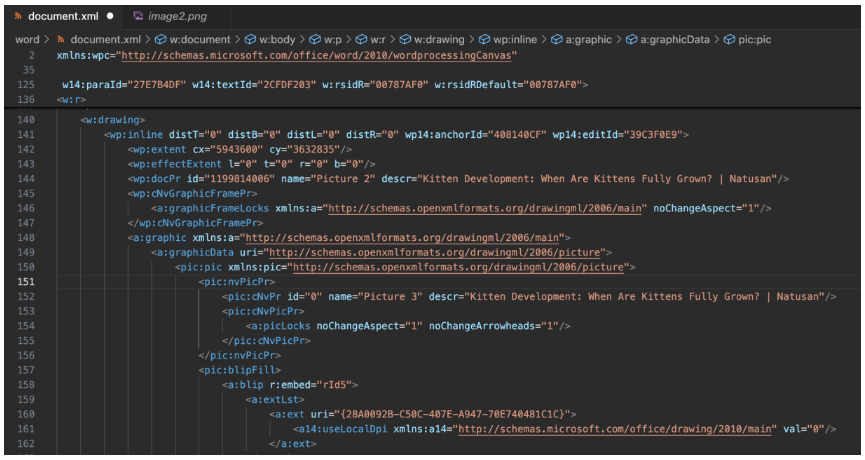
- **Analysis Performed:**
  - The document.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application
  - The examiner opened the document.xml file within Visual Studio Code to inspect embedded media content.
  - Two image entries were located: Picture 1 ("San Antonio Pets Alive") and Picture 2 ("Kitten Development: When Are Kittens Fully Grown?") as shown in Figures 13 and 14.
  - Each image reference was confirmed through unique r:embed IDs (rId4 and rId5) under the <a:blip> tag.
- **Answer:**
  The document.xml file shows **two embedded images**, identified by the <wp:docPr> elements with names "Picture 1" and "Picture 2", as shown in Figures 13 and 14.

- **Supporting Evidence:**



```
document.xml
word > document.xml > w:document > w:body > w:p > w:r > w:rPr > w:noProof
ment>
ody>
<w:p w14:paraId="789E0D96" w14:textId="34BE3B9A" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
    <w:r>
        <w:rPr>
        </w:rPr>
        <w:drawing>
            <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="39746F32" wp14:editId="1E4413F4">
                <wp:extent cx="3304540" cy="2465705"/>
                <wp:effectExtent l="0" t="0" r="0" b="0"/>
                <wp:docPr id="1015846224" name="Picture 1" descr="San Antonio Pets Alive (Cats &amp; Kittens) | San Antonio TX"/>
                <wp:cNvGraphicFramePr>
                    <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
                </wp:cNvGraphicFramePr>
                <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                        <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                            <pic:nvPicPr>
                                <pic:cNvPr id="0" name="Picture 1" descr="San Antonio Pets Alive (Cats &amp; Kittens) | San Antonio TX"/>
                                <pic:cNvPicPr>
                                    <a:picLocks noChangeAspect="1" noChangeArrowheads="1"/>
                                </pic:cNvPicPr>
                            </pic:nvPicPr>
                            <pic:blipFill>
                                <a:blip r:embed="rId4">
                                    <a:extLst>
                                        <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                            <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
```

Figure 13. Screenshot of document.xml showing the first embedded image, labeled "Picture 1 – San Antonio Pets Alive (Cats & Kittens) | San Antonio TX".



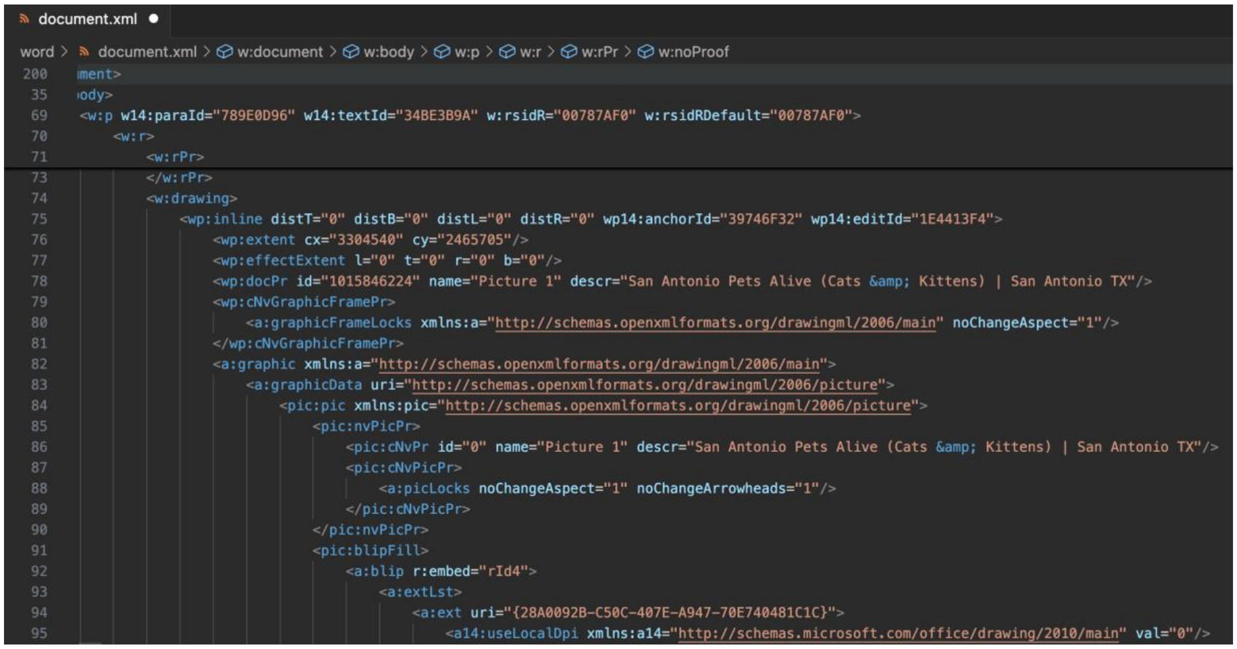
```
document.xml    image2.png
word > document.xml > w:document > w:body > w:p > w:r > w:drawing > wp:inline > a:graphic > a:graphicData > pic:pic
xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"

 w14:paraId="27E7B4DF" w14:textId="2CFDF203" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
<w:r>
    <w:drawing>
        <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="408140CF" wp14:editId="39C3F0E9">
            <wp:extent cx="5943600" cy="3632835"/>
            <wp:effectExtent l="0" t="0" r="0" b="0"/>
            <wp:docPr id="1199814006" name="Picture 2" descr="Kitten Development: When Are Kittens Fully Grown? | Natusan"/>
            <wp:cNvGraphicFramePr>
                <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
            </wp:cNvGraphicFramePr>
            <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                    <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                        <pic:nvPicPr>
                            <pic:cNvPr id="0" name="Picture 3" descr="Kitten Development: When Are Kittens Fully Grown? | Natusan"/>
                            <pic:cNvPicPr>
                                <a:picLocks noChangeAspect="1" noChangeArrowheads="1"/>
                            </pic:cNvPicPr>
                        </pic:nvPicPr>
                        <pic:blipFill>
                            <a:blip r:embed="rId5">
                                <a:extLst>
                                    <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                        <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
                                    </a:ext>
```

Figure 14. Screenshot of document.xml showing the second embedded image, labeled "Picture 2 – Kitten Development: When Are Kittens Fully Grown? | Natusan".

8. What is the 10-digit ID and the description (descr) of each image?

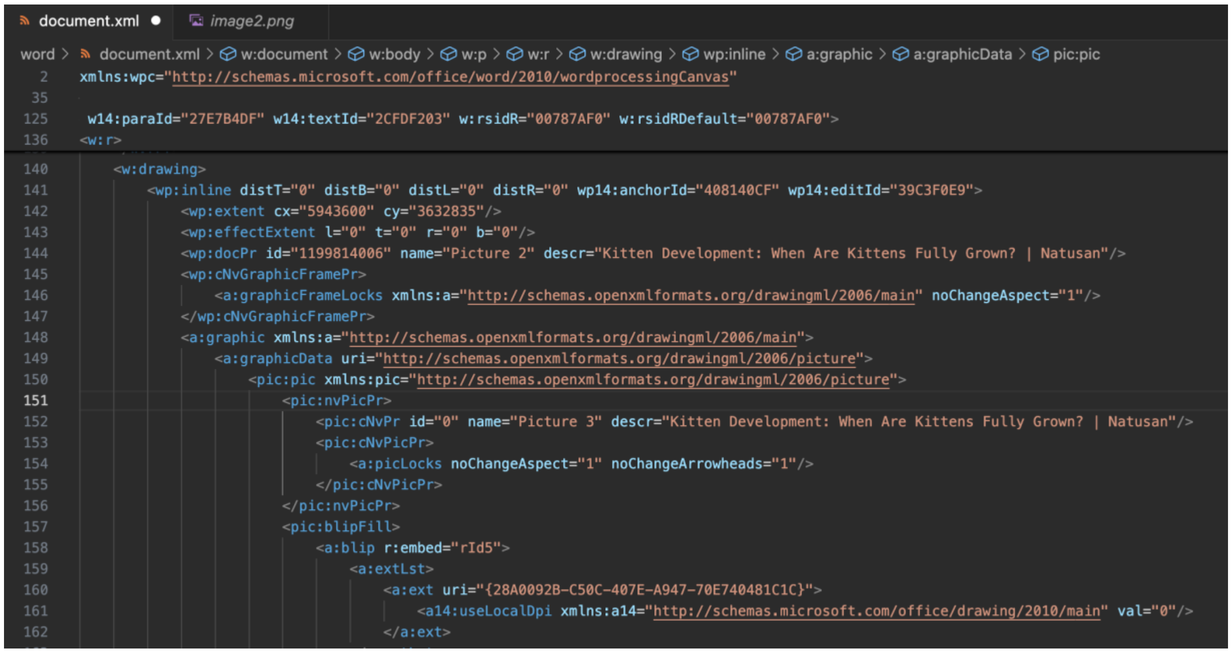
- **Analysis Performed:**
  - The document.docx document file was examined through the examiner’s macOS host computer’s Visual Studio Code application
  - The examiner opened the document.xml file within Visual Studio Code to inspect embedded media content.
  - Two image entries were located: Picture 1 and Picture 2, as shown in Figures 13 and 14.
  - The examiner reviewed the <wp:docPr> elements within document.xml to extract each image’s unique 10-digit ID and corresponding description (descr) attribute, as shown in Figures 15 and 16.
  - The 10-digit ID uniquely identifies each embedded image within the Word document, allowing for precise referencing, linking, and management of media files in the OOXML package.
- **Answer:**
  For first image (Picture 1), the 10-digit ID is “**1015826424**” and for the second image (Picture 2), the 10-digit ID is “**1199814006**”, all shown in Figures 15 and 16. For the first image (Picture 1), the description is: “**San Antonio Pets Alive (Cats & Kittens) | San Antonio TX**”, as shown in Figures 15 and 16. For the second image (Picture 2), the description is: “**Kitten Development: When Are Kittens Fully Grown? | Natusan**”, as shown in Figures 15 and 16.

- **Supporting Evidence:**


document.xml
word > document.xml > w:document > w:body > w:p > w:r > w:rPr > w:noProof

```
ment>
ody>
<w:p w14:paraId="789E0D96" w14:textId="34BE3B9A" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
    <w:r>
        <w:rPr>
        </w:rPr>
        <w:drawing>
            <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="39746F32" wp14:editId="1E4413F4">
                <wp:extent cx="3304540" cy="2465705"/>
                <wp:effectExtent l="0" t="0" r="0" b="0"/>
                <wp:docPr id="1015846224" name="Picture 1" descr="San Antonio Pets Alive (Cats &amp; Kittens) | San Antonio TX"/>
                <wp:cNvGraphicFramePr>
                    <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
                </wp:cNvGraphicFramePr>
                <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                        <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                            <pic:nvPicPr>
                                <pic:cNvPr id="0" name="Picture 1" descr="San Antonio Pets Alive (Cats &amp; Kittens) | San Antonio TX"/>
                                <pic:cNvPicPr>
                                    <a:picLocks noChangeAspect="1" noChangeArrowheads="1"/>
                                </pic:cNvPicPr>
                            </pic:nvPicPr>
                            <pic:blipFill>
                                <a:blip r:embed="rId4">
                                    <a:extLst>
                                        <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                            <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
```

Figure 15. Screenshot of document.xml showing the first embedded image, labeled “Picture 1 – San Antonio Pets Alive (Cats & Kittens) | San Antonio TX”.



document.xml
image2.png
word > document.xml > w:document > w:body > w:p > w:r > w:drawing > wp:inline > a:graphic > a:graphicData > pic:pic

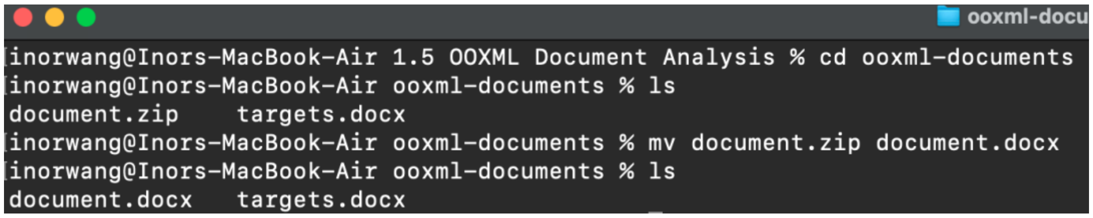
```
xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"

 w14:paraId="27E7B4DF" w14:textId="2CFDF203" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
<w:r>
    <w:drawing>
        <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="408140CF" wp14:editId="39C3F0E9">
            <wp:extent cx="5943600" cy="3632835"/>
            <wp:effectExtent l="0" t="0" r="0" b="0"/>
            <wp:docPr id="1199814006" name="Picture 2" descr="Kitten Development: When Are Kittens Fully Grown? | Natusan"/>
            <wp:cNvGraphicFramePr>
                <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
            </wp:cNvGraphicFramePr>
            <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                    <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                        <pic:nvPicPr>
                            <pic:cNvPr id="0" name="Picture 3" descr="Kitten Development: When Are Kittens Fully Grown? | Natusan"/>
                            <pic:cNvPicPr>
                                <a:picLocks noChangeAspect="1" noChangeArrowheads="1"/>
                            </pic:cNvPicPr>
                        </pic:nvPicPr>
                        <pic:blipFill>
                            <a:blip r:embed="rId5">
                                <a:extLst>
                                    <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                        <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
                                    </a:ext>
```

Figure 16. Screenshot of document.xml showing the second embedded image, labeled "Picture 2 – Kitten Development: When Are Kittens Fully Grown? | Natusan".

9. How many images are really in this document?

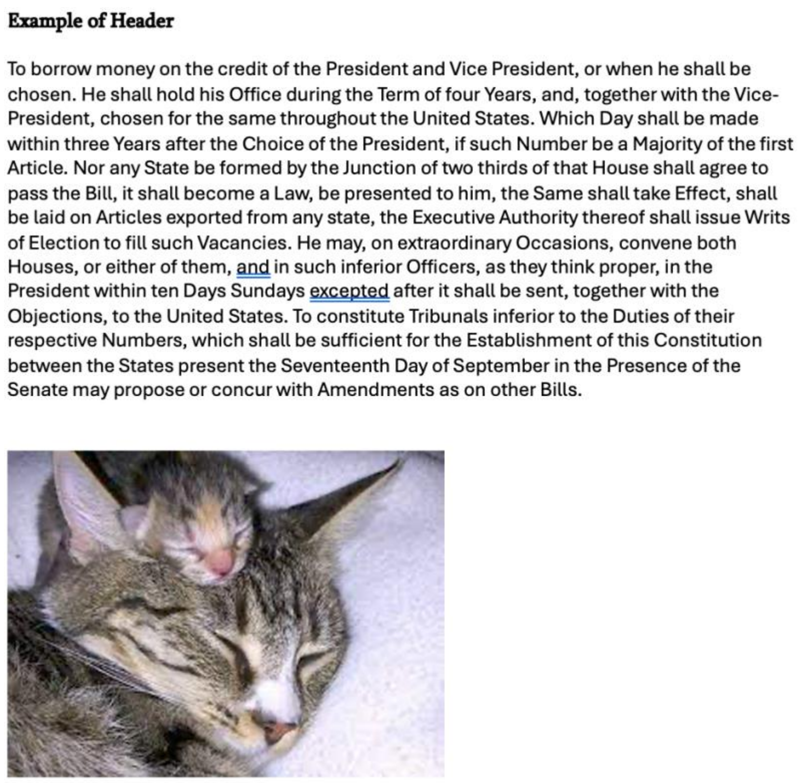
- **Analysis Performed:**
    - The document.docx document file was examined through the examiner's macOS host computer's terminal and Microsoft Word application.
    - The examiner renamed document.zip back to document.docx using the mv command to restore the file to its original Word document format, as shown in Figure 17.
        - Command: "*mv document.zip document.docx*"
    - From there, the examiner accessed the document.
- **Answer:**
  There are **two images really in this document,** as shown in Figures 18 and 19.

- **Supporting Evidence:**



```
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % cd ooxml-documents
inorwang@Inors-MacBook-Air ooxml-documents % ls
document.zip    targets.docx
inorwang@Inors-MacBook-Air ooxml-documents % mv document.zip document.docx
inorwang@Inors-MacBook-Air ooxml-documents % ls
document.docx   targets.docx
```

Figure 17. Screenshot showing the examiner renaming document.zip back to document.docx after analysis to restore the original file format.



Example of Header

To borrow money on the credit of the President and Vice President, or when he shall be chosen. He shall hold his Office during the Term of four Years, and, together with the Vice-President, chosen for the same throughout the United States. Which Day shall be made within three Years after the Choice of the President, if such Number be a Majority of the first Article. Nor any State be formed by the Junction of two thirds of that House shall agree to pass the Bill, it shall become a Law, be presented to him, the Same shall take Effect, shall be laid on Articles exported from any state, the Executive Authority thereof shall issue Writs of Election to fill such Vacancies. He may, on extraordinary Occasions, convene both Houses, or either of them, and in such inferior Officers, as they think proper, in the President within ten Days Sundays excepted after it shall be sent, together with the Objections, to the United States. To constitute Tribunals inferior to the Duties of their respective Numbers, which shall be sufficient for the Establishment of this Constitution between the States present the Seventeenth Day of September in the Presence of the Senate may propose or concur with Amendments as on other Bills.

Figure 18. First page of the document.docx file

Figure 19. Second page of the document.docx file

10. If additional images are found, what are the filenames and describe the contents.

- **Analysis Performed:**
  - The document.docx document file was examined through the examiner's macOS host computer's terminal and Microsoft Word application.
  - The examiner navigated to the word/media directory within the extracted OOXML structure to inspect all embedded image files.
  - Additional images were identified beyond those referenced in document.xml, including image3.png.
  - Each image file was manually opened to verify its visual contents and confirm successful extraction.
  - image3.png was found to depict a dog on grass wearing a pink plaid outfit, indicating it was an embedded media object not directly referenced in the main XML body.
  - As shown in Figure 21, there was an INCLUDEPICTURE reference to an external image.
- **Answer:**
  Upon further inspection of the word/media directory, an additional image file named image3.png was discovered, as shown in Figure 20. This image was not directly referenced within the document.xml file but was still embedded within the document package. The examiner reviewed the file visually and confirmed that image3.png depicts a dog standing on grass, wearing a pink plaid outfit. This finding indicates that the Word document contained at least one additional embedded image beyond those identified in the main XML structure.

- **Supporting Evidence:**

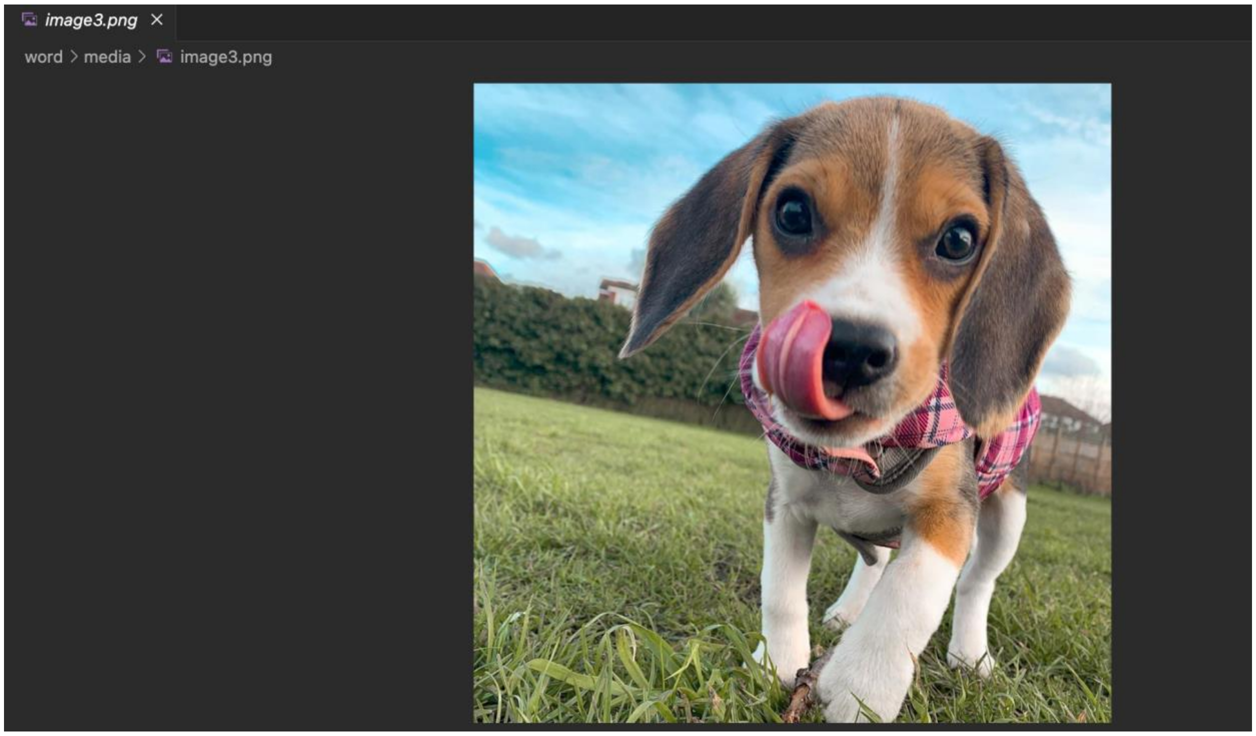
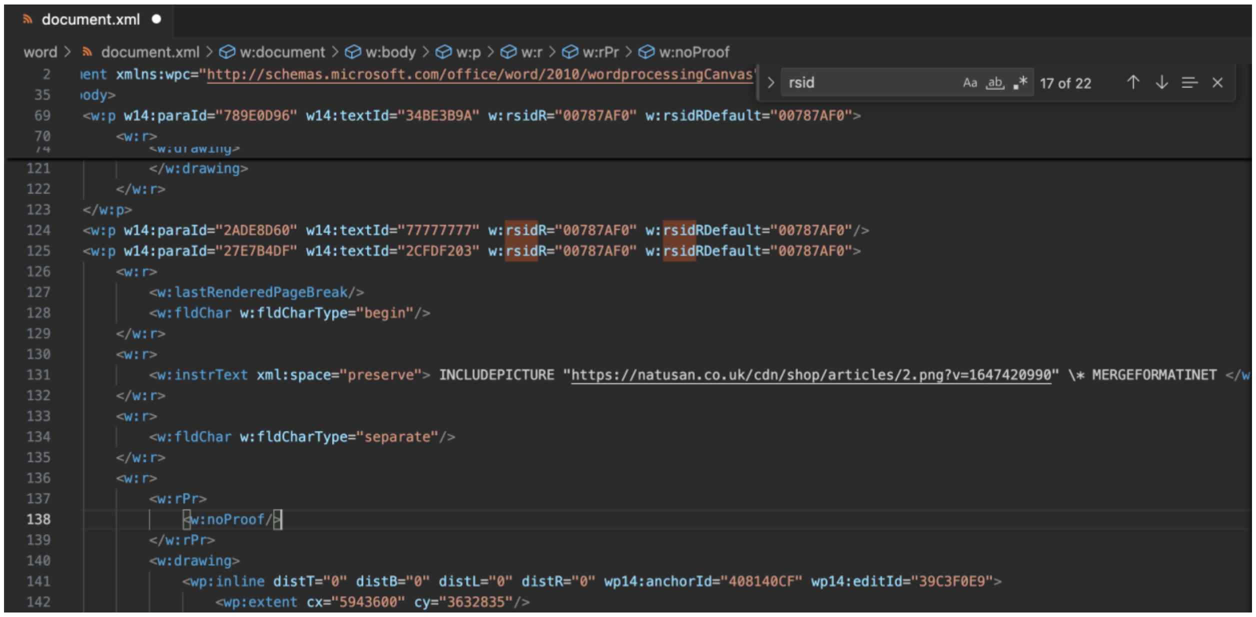


Figure 20. Screenshot showing image3.png extracted from the Word document's word/media folder, displaying a dog image embedded within the document.


document.xml
word > document.xml > w:document > w:body > w:p > w:r > w:rPr > w:noProof
rsid
Aa ab .*
17 of 22

```
ıent xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas
ıody>
<w:p w14:paraId="789E0D96" w14:textId="34BE3B9A" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
    <w:r>
        <w:drawing>
        </w:drawing>
    </w:r>
</w:p>
<w:p w14:paraId="2ADE8D60" w14:textId="77777777" w:rsidR="00787AF0" w:rsidRDefault="00787AF0"/>
<w:p w14:paraId="27E7B4DF" w14:textId="2CFDF203" w:rsidR="00787AF0" w:rsidRDefault="00787AF0">
    <w:r>
        <w:lastRenderedPageBreak/>
        <w:fldChar w:fldCharType="begin"/>
    </w:r>
    <w:r>
        <w:instrText xml:space="preserve"> INCLUDEPICTURE "https://natusan.co.uk/cdn/shop/articles/2.png?v=1647420990" \* MERGEFORMATINET </w
    </w:r>
    <w:r>
        <w:fldChar w:fldCharType="separate"/>
    </w:r>
    <w:r>
        <w:rPr>
            <w:noProof/>
        </w:rPr>
        <w:drawing>
            <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="408140CF" wp14:editId="39C3F0E9">
                <wp:extent cx="5943600" cy="3632835"/>
```

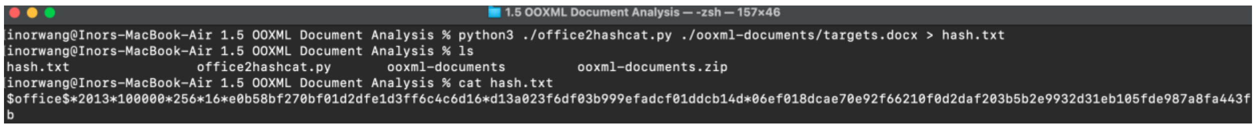
Figure 21. INCLUDEPICTURE line within document.xml of document.docx

targets.docx

1. What is the output from office2hashcat?

- **Analysis Performed:**
    - The targets.docx document file was examined through the examiner's macOS host computer's terminal.
    - The examiner downloaded the "office2hashcat.py" python file from the Professor's Canvas webpage.
    - The examiner executed the office2hashcat.py script to extract the password hash from the encrypted targets.docx file, redirecting the output to a text file named hash.txt.
        - Command: "*python3 ./office2hashcat.py ./ooxml-documents/targets.docx > hash.txt*"
    - The script successfully produced a hash in Hashcat-compatible format, beginning with "$office$*2013*...", indicating that the file uses Microsoft Office 2013 encryption, as shown in Figure 22.
    - The output string in hash.txt contains the parameters and encoded hash data required for password recovery using Hashcat.
- **Answer:**
  The examiner ran the office2hashcat.py tool against the encrypted targets.docx and redirected the output into hash.txt. **The script produced a Hashcat-compatible extraction string (beginning with $office$*2013*…) that encodes the Office 2013 encryption parameters (salt, iteration counts, and ciphertext).** This hash.txt file now contains the necessary hash data and options required to attempt password recovery with Hashcat.

- **Supporting Evidence:**



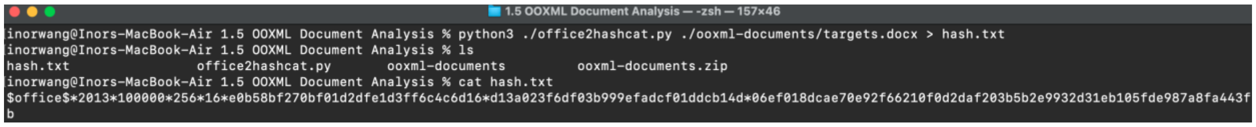
```
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % python3 ./office2hashcat.py ./ooxml-documents/targets.docx > hash.txt
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % ls
hash.txt                office2hashcat.py       ooxml-documents         ooxml-documents.zip
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % cat hash.txt
$office$*2013*100000*256*16*e0b58bf270bf01d2dfe1d3ff6c4c6d16*d13a023f6df03b999efadcf01ddcb14d*06ef018dcae70e92f66210f0d2daf203b5b2e9932d31eb105fde987a8fa443f
b
```

Figure 22. Screenshot showing the examiner extracting the password hash from targets.docx using the office2hashcat.py script, outputting the result to hash.txt for further password recovery analysis.

2. What version of Office does this document use?

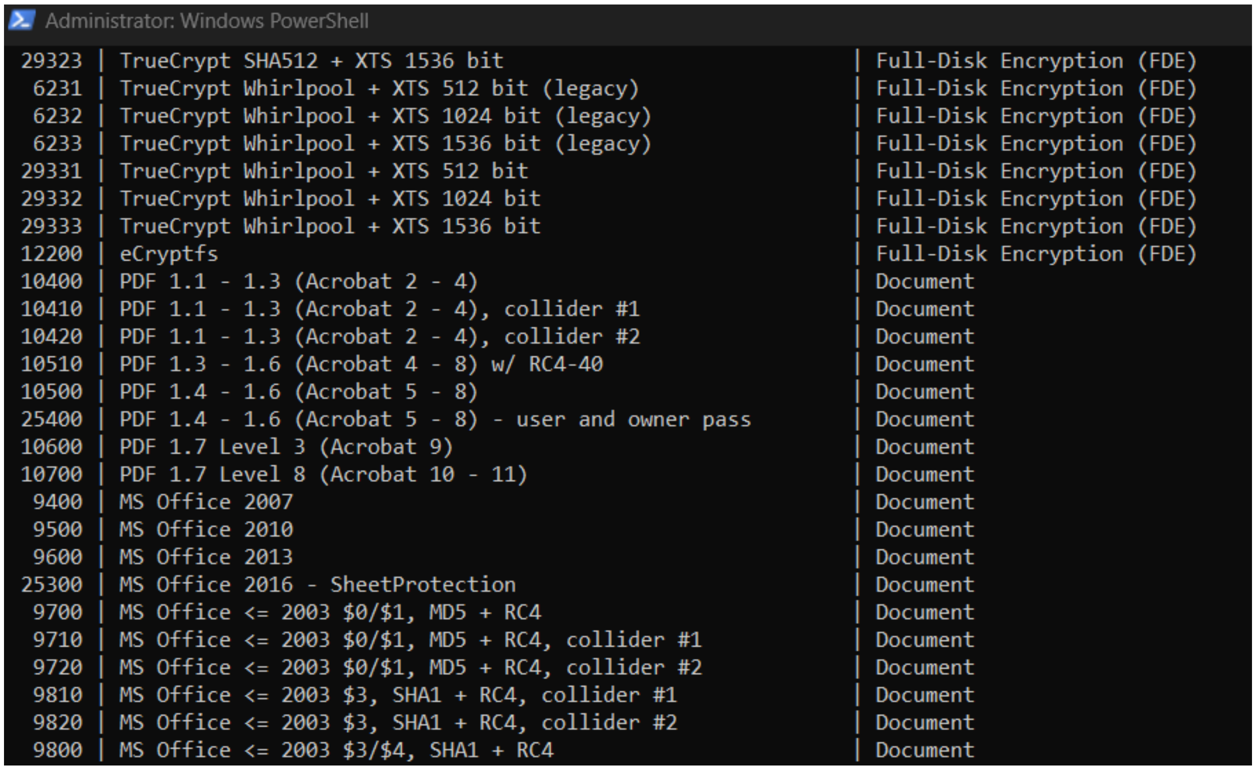
- **Analysis Performed:**
  - From the output of the office2hashcat.py script as shown previously and in Figure 23, the version of Office that the document, targets.docx, uses is Office 2013.
  - From there, the examiner used hashcat to determine which mode to use to crack the password.
    - Command: ./hashcat.exe -h
    - The correct mode for this document, targets.docx, is "**9600**", as shown in Figure 24.
- **Answer:**
  The version of Office that this document, targets.docx, uses is **Office 2013** as shown in Figure 23.

- **Supporting Evidence:**



```
1.5 OOXML Document Analysis — -zsh — 157×46
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % python3 ./office2hashcat.py ./ooxml-documents/targets.docx > hash.txt
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % ls
hash.txt                office2hashcat.py       ooxml-documents         ooxml-documents.zip
inorwang@Inors-MacBook-Air 1.5 OOXML Document Analysis % cat hash.txt
$office$*2013*100000*256*16*e0b58bf270bf01d2dfe1d3ff6c4c6d16*d13a023f6df03b999efadcf01ddcb14d*06ef018dcae70e92f66210f0d2daf203b5b2e9932d31eb105fde987a8fa443f
b
```

Figure 23. Screenshot showing the examiner extracting the password hash from targets.docx using the office2hashcat.py script, outputting the result to hash.txt for further password recovery analysis.



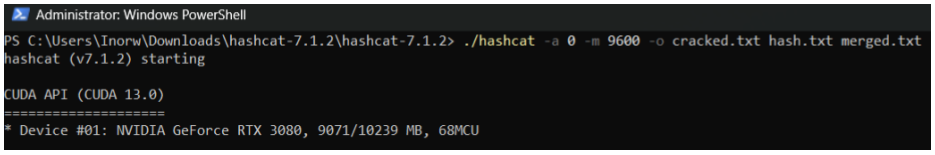
```
Administrator: Windows PowerShell
 29323 | TrueCrypt SHA512 + XTS 1536 bit                             | Full-Disk Encryption (FDE)
  6231 | TrueCrypt Whirlpool + XTS 512 bit (legacy)                  | Full-Disk Encryption (FDE)
  6232 | TrueCrypt Whirlpool + XTS 1024 bit (legacy)                 | Full-Disk Encryption (FDE)
  6233 | TrueCrypt Whirlpool + XTS 1536 bit (legacy)                 | Full-Disk Encryption (FDE)
 29331 | TrueCrypt Whirlpool + XTS 512 bit                           | Full-Disk Encryption (FDE)
 29332 | TrueCrypt Whirlpool + XTS 1024 bit                          | Full-Disk Encryption (FDE)
 29333 | TrueCrypt Whirlpool + XTS 1536 bit                          | Full-Disk Encryption (FDE)
 12200 | eCryptfs                                                    | Full-Disk Encryption (FDE)
 10400 | PDF 1.1 - 1.3 (Acrobat 2 - 4)                               | Document
 10410 | PDF 1.1 - 1.3 (Acrobat 2 - 4), collider #1                  | Document
 10420 | PDF 1.1 - 1.3 (Acrobat 2 - 4), collider #2                  | Document
 10510 | PDF 1.3 - 1.6 (Acrobat 4 - 8) w/ RC4-40                     | Document
 10500 | PDF 1.4 - 1.6 (Acrobat 5 - 8)                               | Document
 25400 | PDF 1.4 - 1.6 (Acrobat 5 - 8) - user and owner pass         | Document
 10600 | PDF 1.7 Level 3 (Acrobat 9)                                 | Document
 10700 | PDF 1.7 Level 8 (Acrobat 10 - 11)                           | Document
  9400 | MS Office 2007                                              | Document
  9500 | MS Office 2010                                              | Document
  9600 | MS Office 2013                                              | Document
 25300 | MS Office 2016 - SheetProtection                            | Document
  9700 | MS Office <= 2003 $0/$1, MD5 + RC4                          | Document
  9710 | MS Office <= 2003 $0/$1, MD5 + RC4, collider #1             | Document
  9720 | MS Office <= 2003 $0/$1, MD5 + RC4, collider #2             | Document
  9810 | MS Office <= 2003 $3, SHA1 + RC4, collider #1               | Document
  9820 | MS Office <= 2003 $3, SHA1 + RC4, collider #2               | Document
  9800 | MS Office <= 2003 $3/$4, SHA1 + RC4                         | Document
```

Figure 24. Figure 13. Screenshot of Hashcat's supported hash modes, highlighting mode 9600 (MS Office 2013) — the algorithm used for the encrypted targets.docx file.

3. What is the full hashcat command you used to crack this password hash?

- **Analysis Performed:**
  - The hash.txt file that the examiner received from the office2hashcat was analyzed through the hashcat application.
  - The examiner extracted the Office hash and executed Hashcat in straight/dictionary mode (-a 0, mode 9600) on an NVIDIA RTX 3080, directing results to cracked.txt and logging attack parameters and performance for reproducibility.
- **Answer:**
  The full hashcat command the examiner used to crack the password hash is: "**./hashcat -a 0 -m 9600 -o cracked.txt hash.txt merged.txt**", as shown in Figure 25. The mode was determined in the previous question in Figure 24.

- **Supporting Evidence:**



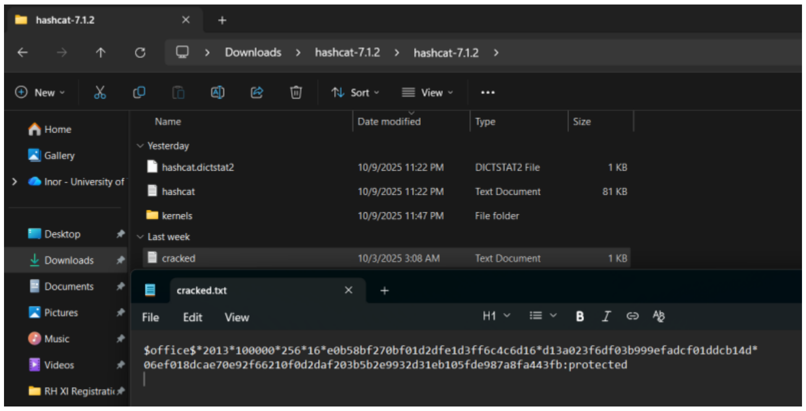
```
PS C:\Users\Inorw\Downloads\hashcat-7.1.2\hashcat-7.1.2> ./hashcat -a 0 -m 9600 -o cracked.txt hash.txt merged.txt
hashcat (v7.1.2) starting

CUDA API (CUDA 13.0)
====================
* Device #01: NVIDIA GeForce RTX 3080, 9071/10239 MB, 68MCU
```

Figure 25. Execution of Hashcat (v7.1.2) in Windows PowerShell

4. What was the cracked password?

- **Analysis Performed:**
  - The cracked.txt file was created whenever the hashcat command finished. The examiner accessed the .txt file to acquire the cracked password.
  - The recovered plaintext "protected" was verified in cracked.txt, confirming successful decryption of the document and validating that the file was originally password-protected.
- **Answer:**
  The cracked password of targets.docx is "**protected**", as shown in Figure 26.

- **Supporting Evidence:**



```
Name                 Date modified        Type             Size
Yesterday
hashcat.dictstat2    10/9/2025 11:22 PM   DICTSTAT2 File   1 KB
hashcat              10/9/2025 11:22 PM   Text Document    81 KB
kernels              10/9/2025 11:47 PM   File folder
Last week
cracked              10/3/2025 3:08 AM    Text Document    1 KB
```

```
$office$*2013*100000*256*16*e0b58bf270bf01d2dfe1d3ff6c4c6d16*d13a023f6df03b999efadcf01ddcb14d*
06ef018dcae70e92f66210f0d2daf203b5b2e9932d31eb105fde987a8fa443fb:protected
```

Figure 26. Cracked password of targets.docx in cracked.txt

5. Open the document using the password. Describe the contents.

- **Analysis Performed:**
    - The targets.docx document file was examined through the examiner's macOS host computer's Microsoft Word application.
    - The examiner accessed the document using the password, as shown in Figure 27.
- **Answer:**
  The targets.docx document page, labeled "FOR TRAINING PURPOSES ONLY" at both the top and bottom, contains three images labeled Target #1, Target #2, and Target #3. Target #1 depicts a scenic landscape with greenery, trees in autumn colors, and a small structure or house in the background. Target #2 shows a park-like setting with a large tree in the foreground, a bench, manicured bushes, and a white monument or fountain in the distance. Target #3 features a narrow urban street bordered by tall buildings on either side, with a few pedestrians visible along the pathway. The images are arranged vertically on the page for clear comparison or reference.

- **Supporting Evidence:**

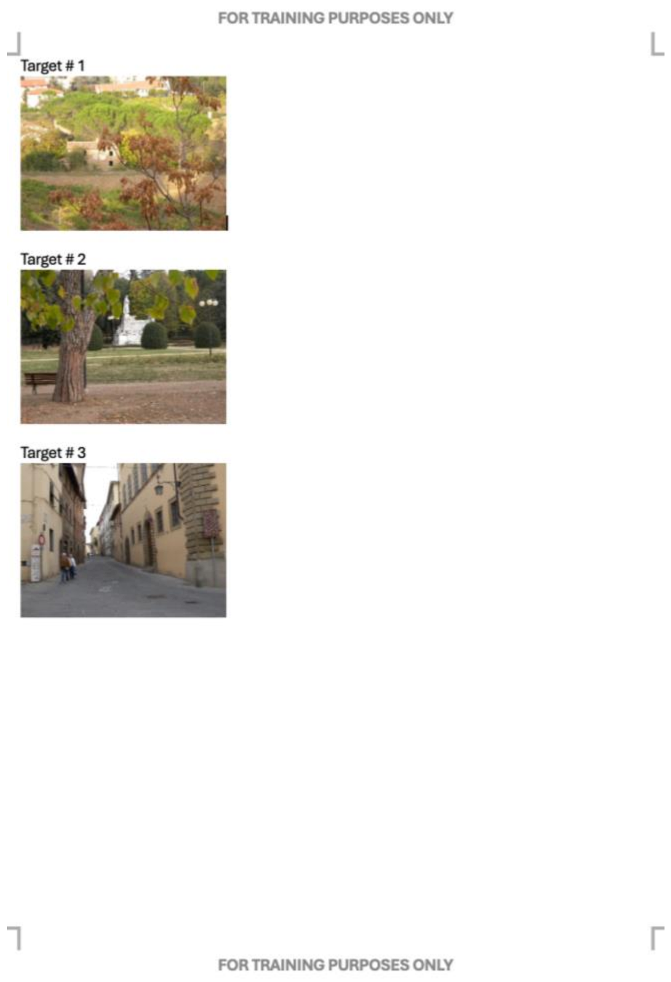
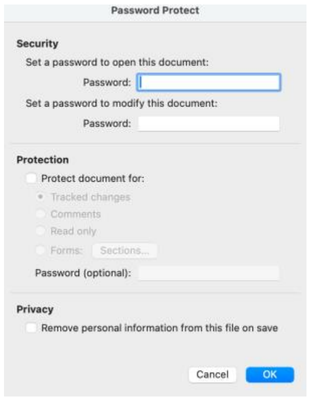


Figure 27. Contents of targets.docx

6. Who is the creator of this document?

- **Analysis Performed:**
  - The targets.docx document file was examined through the examiner's macOS host computer's Microsoft Word application, terminal, and Visual Studio Code application.
  - The examiner opened targets.docx in Microsoft Word, navigated to the Review tab, and accessed the Password Protect settings to remove the document's password, effectively disabling encryption and allowing unrestricted access to the file's contents, as shown in Figure 28.
  - The .docx file was renamed to .zip and extracted using the unzip command, revealing its OOXML (Open Office XML) structure as shown in Figure 29.
    - Commands: "*mv targets.docx targets.zip*" and "*unzip targets.zip*"
  - The examiner formatted the core.xml file to easily analyze the contents, which contained who the creator of the document is ("Guy Faux"), as shown in Figure 30.
- **Answer:**
  The creator of the targets.docx file is "**Guy Faux**", as shown in Figure 30.

- **Supporting Evidence:**


Password Protect

Security
Set a password to open this document:
Password:
Set a password to modify this document:
Password:

Protection
Protect document for:
Tracked changes
Comments
Read only
Forms: Sections...
Password (optional):

Privacy
Remove personal information from this file on save

Cancel OK

Figure 28. Removing the password protection for targets.docx

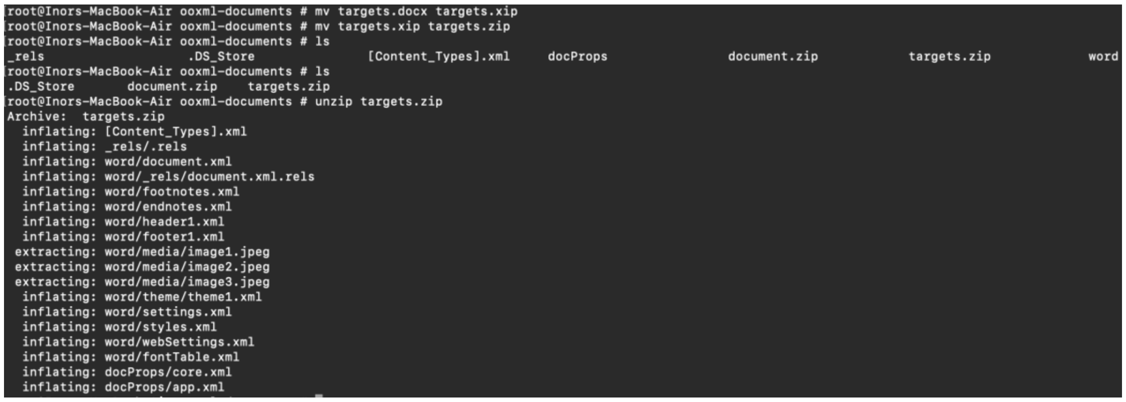
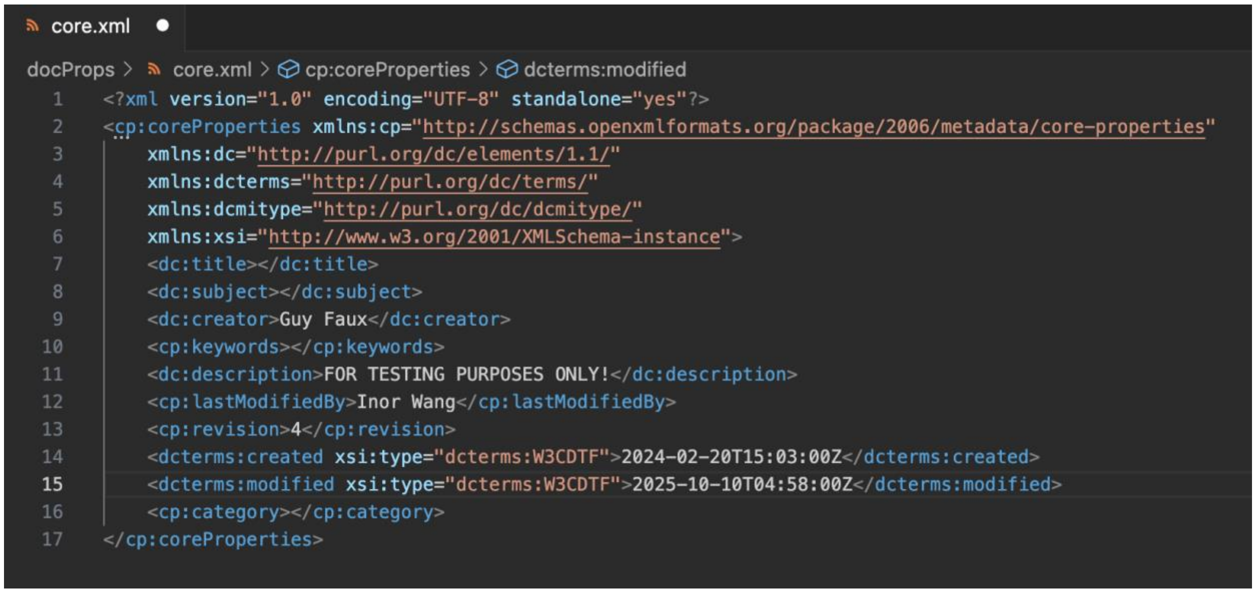

```
root@Inors-MacBook-Air ooxml-documents # mv targets.docx targets.xip
root@Inors-MacBook-Air ooxml-documents # mv targets.xip targets.zip
root@Inors-MacBook-Air ooxml-documents # ls
_rels                  .DS_Store              [Content_Types].xml    docProps               document.zip           targets.zip            word
root@Inors-MacBook-Air ooxml-documents # ls
.DS_Store       document.zip    targets.zip
root@Inors-MacBook-Air ooxml-documents # unzip targets.zip
Archive:  targets.zip
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: word/document.xml
  inflating: word/_rels/document.xml.rels
  inflating: word/footnotes.xml
  inflating: word/endnotes.xml
  inflating: word/header1.xml
  inflating: word/footer1.xml
 extracting: word/media/image1.jpeg
 extracting: word/media/image2.jpeg
 extracting: word/media/image3.jpeg
  inflating: word/theme/theme1.xml
  inflating: word/settings.xml
  inflating: word/styles.xml
  inflating: word/webSettings.xml
  inflating: word/fontTable.xml
  inflating: docProps/core.xml
  inflating: docProps/app.xml
```

Figure 29. Screenshot showing the examiner renaming targets.docx to targets.zip and extracting its contents, revealing the internal OOXML structure including XML files and embedded media images.



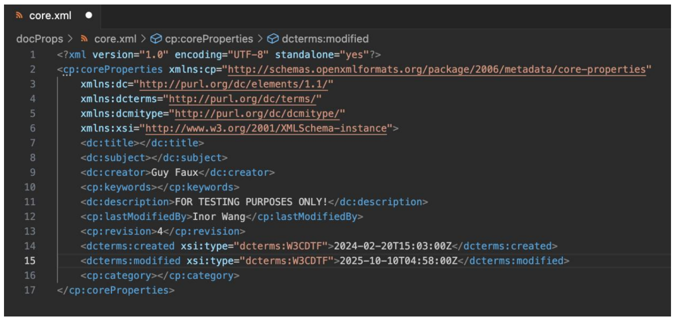
```
core.xml
docProps > core.xml > cp:coreProperties > dcterms:modified
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Faux</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description>FOR TESTING PURPOSES ONLY!</dc:description>
    <cp:lastModifiedBy>Inor Wang</cp:lastModifiedBy>
    <cp:revision>4</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T15:03:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2025-10-10T04:58:00Z</dcterms:modified>
    <cp:category></cp:category>
</cp:coreProperties>
```

Figure 30. Screenshot of core.xml opened in Visual Studio Code showing document metadata for targets.docx

7. Who last modified this document?

- **Analysis Performed:**
  - The targets.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
  - The examiner formatted the core.xml file to easily analyze the contents, which contained who last modified the document ("Inor Wang"), as shown in Figure 31.
- **Answer:**
  The person who last modified the targets.docx file was "**Inor Wang**", as shown in Figure 31. The reason for this is because in order to unzip the target.docx file to extract the XML internal structure, it cannot be password-protected. Therefore, as stated previously, the examiner, Inor Wang, removed the password and saved the document in order to nullify the previous password-protection.

- **Supporting Evidence:**



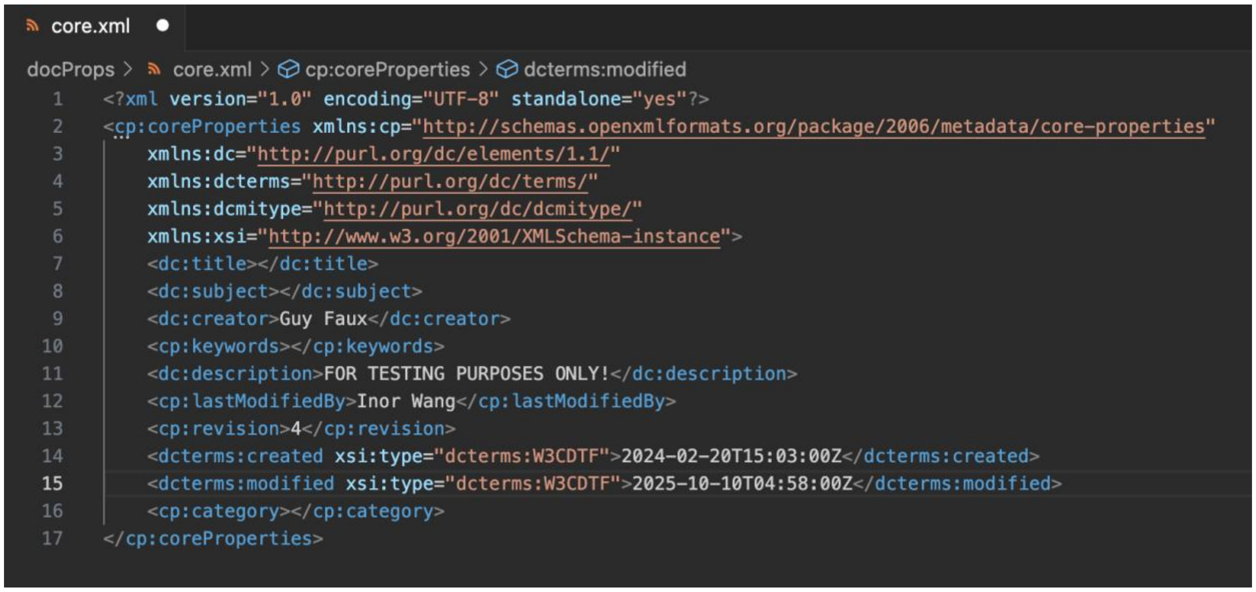
```
core.xml
docProps > core.xml > cp:coreProperties > dcterms:modified
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Faux</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description>FOR TESTING PURPOSES ONLY!</dc:description>
    <cp:lastModifiedBy>Inor Wang</cp:lastModifiedBy>
    <cp:revision>4</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T15:03:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2025-10-10T04:58:00Z</dcterms:modified>
    <cp:category></cp:category>
</cp:coreProperties>
```

Figure 31. Screenshot of core.xml opened in Visual Studio Code showing document metadata for targets.docx

8. When was this document created?

- **Analysis Performed:**
    - The targets.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
    - The examiner formatted the core.xml file to easily analyze the contents, which contained when the targets.docx was created ("2024-02-20T15:03:00Z"), as shown in Figure 32.
    - The timestamp ends with "Z", indicating the time is recorded in Coordinated Universal Time (UTC).
- **Answer:**
  The targets.docx file was created on **February 20, 2024, at 15:03:00 UTC**, as shown in Figure 32.

- **Supporting Evidence:**



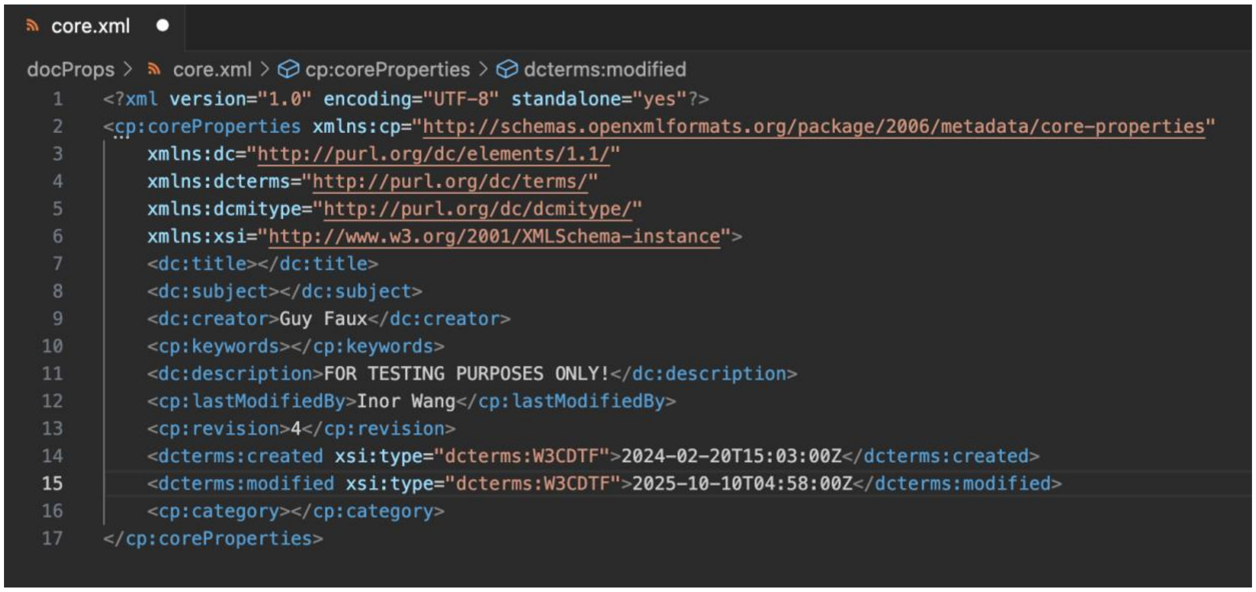
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Faux</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description>FOR TESTING PURPOSES ONLY!</dc:description>
    <cp:lastModifiedBy>Inor Wang</cp:lastModifiedBy>
    <cp:revision>4</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T15:03:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2025-10-10T04:58:00Z</dcterms:modified>
    <cp:category></cp:category>
</cp:coreProperties>
```

Figure 32. Screenshot of core.xml opened in Visual Studio Code showing document metadata for targets.docx

9. When was this document last modified?

- **Analysis Performed:**
    - The targets.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
    - The examiner formatted the core.xml file to easily analyze the contents, which contained when the targets.docx was last modified ("2025-10-10T04:58:00Z"), as shown in Figure 33.
    - The timestamp ends with "Z", indicating the time is recorded in Coordinated Universal Time (UTC).
- **Answer:**
  The targets.docx file was last modified on **October 10, 2025, at 04:58:00 UTC**, as shown in Figure 33.

- **Supporting Evidence:**



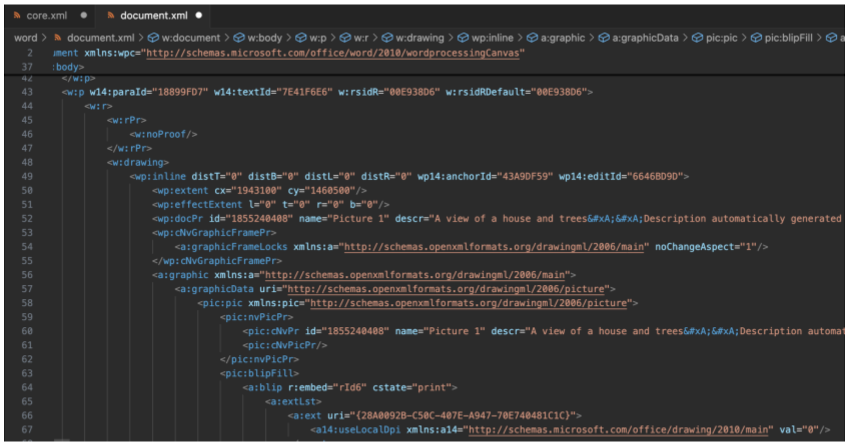
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dc:title></dc:title>
    <dc:subject></dc:subject>
    <dc:creator>Guy Faux</dc:creator>
    <cp:keywords></cp:keywords>
    <dc:description>FOR TESTING PURPOSES ONLY!</dc:description>
    <cp:lastModifiedBy>Inor Wang</cp:lastModifiedBy>
    <cp:revision>4</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2024-02-20T15:03:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2025-10-10T04:58:00Z</dcterms:modified>
    <cp:category></cp:category>
</cp:coreProperties>
```

Figure 33. Screenshot of core.xml opened in Visual Studio Code showing document metadata for targets.docx

10. How many images are in this document?

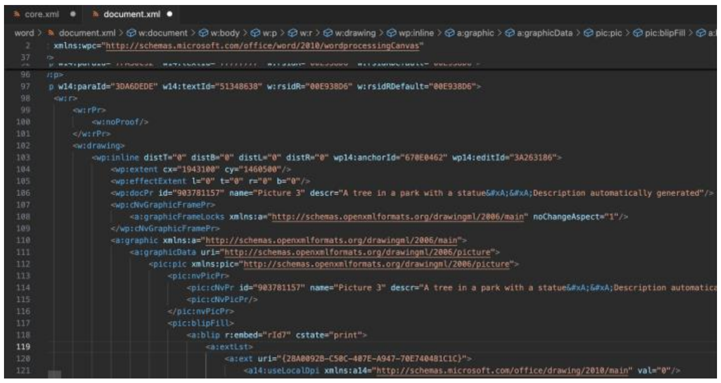
- **Analysis Performed:**
  - The targets.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application and Microsoft Word application.
  - The examiner opened the document.xml file within Visual Studio Code to inspect the embedded media content.
  - Three <wp:drawing> elements were identified in document.xml, labeled Picture 1, Picture 3, and Picture 5, describing "a view of a house and trees," "a tree in a park with a statue," and "a group of people walking down a narrow street," respectively, as shown in Figures 34-36.
  - Each image corresponds to embedded relationship IDs rId6–rId8, linking to /word/media/image1.jpeg, /word/media/image2.jpeg, and /word/media/image3.jpeg as defined in document.xml.rels.
  - Additionally, the examiner accessed the document which content's showed three images, as shown in Figure 37.
- **Answer:**
  There are **3 images** in the targets.docx document file, as shown in Figures 34-37.

- **Supporting Evidence:**



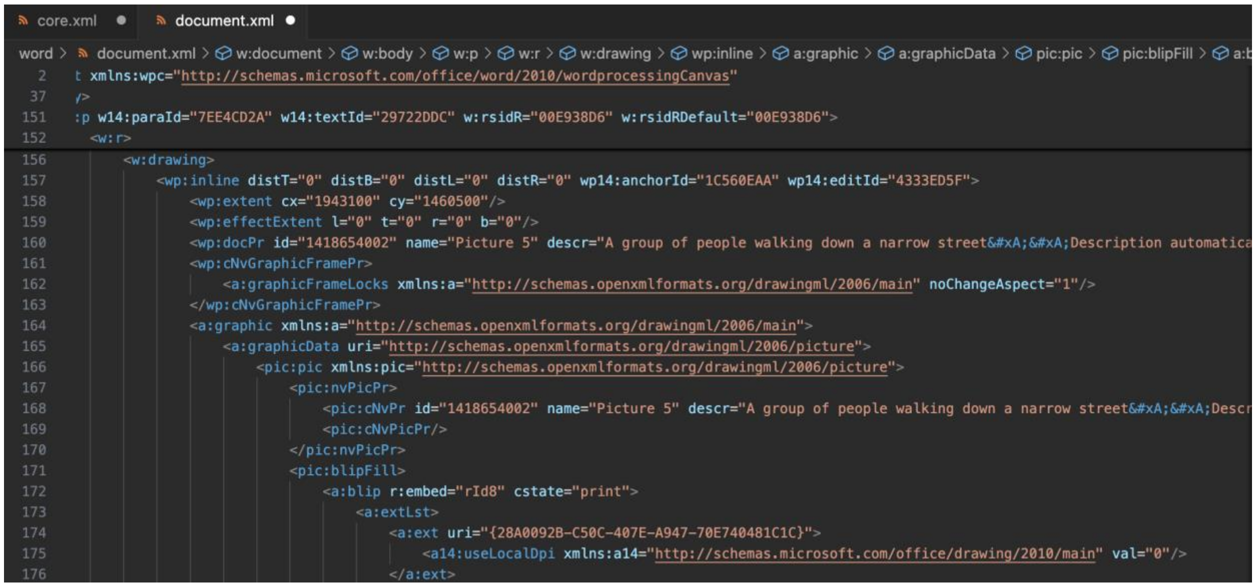
```
core.xml  document.xml
word > document.xml > w:document > w:body > w:p > w:r > w:drawing > wp:inline > a:graphic > a:graphicData > pic:pic > pic:blipFill > a
ıment xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"
:body>
  </w:p>
  <w:p w14:paraId="18899FD7" w14:textId="7E41F6E6" w:rsidR="00E938D6" w:rsidRDefault="00E938D6">
      <w:r>
          <w:rPr>
              <w:noProof/>
          </w:rPr>
          <w:drawing>
              <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="43A9DF59" wp14:editId="6646BD9D">
                  <wp:extent cx="1943100" cy="1460500"/>
                  <wp:effectExtent l="0" t="0" r="0" b="0"/>
                  <wp:docPr id="1855240408" name="Picture 1" descr="A view of a house and trees&#xA;&#xA;Description automatically generated
                  <wp:cNvGraphicFramePr>
                      <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
                  </wp:cNvGraphicFramePr>
                  <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                      <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                          <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                              <pic:nvPicPr>
                                  <pic:cNvPr id="1855240408" name="Picture 1" descr="A view of a house and trees&#xA;&#xA;Description automat
                                  <pic:cNvPicPr/>
                              </pic:nvPicPr>
                              <pic:blipFill>
                                  <a:blip r:embed="rId6" cstate="print">
                                      <a:extLst>
                                          <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                              <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
```

Figure 34. Screenshot of document.xml showing the first embedded image



```
core.xml  document.xml
word > document.xml > w:document > w:body > w:p > w:r > w:drawing > wp:inline > a:graphic > a:graphicData > pic:pic > pic:blipFill > a:
 xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"
r>
/:p>
p w14:paraId="3DA6DEDE" w14:textId="51348638" w:rsidR="00E938D6" w:rsidRDefault="00E938D6">
 <w:r>
     <w:rPr>
         <w:noProof/>
     </w:rPr>
     <w:drawing>
         <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="670E0462" wp14:editId="3A263186">
             <wp:extent cx="1943100" cy="1460500"/>
             <wp:effectExtent l="0" t="0" r="0" b="0"/>
             <wp:docPr id="903781157" name="Picture 3" descr="A tree in a park with a statue&#xA;&#xA;Description automatically generated"/>
             <wp:cNvGraphicFramePr>
                 <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
             </wp:cNvGraphicFramePr>
             <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                 <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                     <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                         <pic:nvPicPr>
                             <pic:cNvPr id="903781157" name="Picture 3" descr="A tree in a park with a statue&#xA;&#xA;Description automatica
                             <pic:cNvPicPr/>
                         </pic:nvPicPr>
                         <pic:blipFill>
                             <a:blip r:embed="rId7" cstate="print">
                                 <a:extLst>
                                     <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                         <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
```

Figure 35. Screenshot of document.xml showing the second embedded image

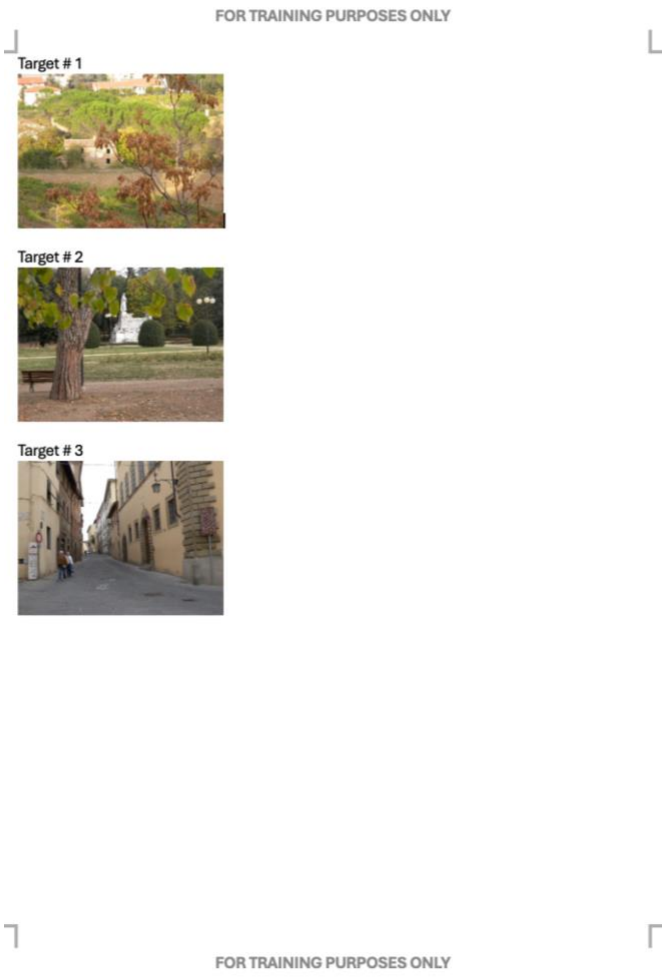


core.xml ● document.xml ●

word > document.xml > w:document > w:body > w:p > w:r > w:drawing > wp:inline > a:graphic > a:graphicData > pic:pic > pic:blipFill > a:b

```xml
t xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"
/>
:p w14:paraId="7EE4CD2A" w14:textId="29722DDC" w:rsidR="00E938D6" w:rsidRDefault="00E938D6">
  <w:r>
      <w:drawing>
          <wp:inline distT="0" distB="0" distL="0" distR="0" wp14:anchorId="1C560EAA" wp14:editId="4333ED5F">
              <wp:extent cx="1943100" cy="1460500"/>
              <wp:effectExtent l="0" t="0" r="0" b="0"/>
              <wp:docPr id="1418654002" name="Picture 5" descr="A group of people walking down a narrow street&#xA;&#xA;Description automatica
              <wp:cNvGraphicFramePr>
                  <a:graphicFrameLocks xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" noChangeAspect="1"/>
              </wp:cNvGraphicFramePr>
              <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                  <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
                      <pic:pic xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
                          <pic:nvPicPr>
                              <pic:cNvPr id="1418654002" name="Picture 5" descr="A group of people walking down a narrow street&#xA;&#xA;Descr
                              <pic:cNvPicPr/>
                          </pic:nvPicPr>
                          <pic:blipFill>
                              <a:blip r:embed="rId8" cstate="print">
                                  <a:extLst>
                                      <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
                                          <a14:useLocalDpi xmlns:a14="http://schemas.microsoft.com/office/drawing/2010/main" val="0"/>
                                      </a:ext>
```

Figure 36. Screenshot of document.xml showing the third embedded image



FOR TRAINING PURPOSES ONLY

Target # 1

Target # 2

Target # 3

FOR TRAINING PURPOSES ONLY

Figure 37. Full contents of the targets.docx file

11. If additional images are found, what are the filenames and describe the contents.

- **Analysis Performed:**
  - The targets.docx document file was examined through the examiner's macOS host computer's Visual Studio Code application.
  - The examiner did not find any evidence of any additional images.
  - In the media folder, it contained the three images found in the previous question.
    - Pictures are shown in Figures 38-40
- **Answer:**
  There are **no additional images found**.

- **Supporting Evidence:**

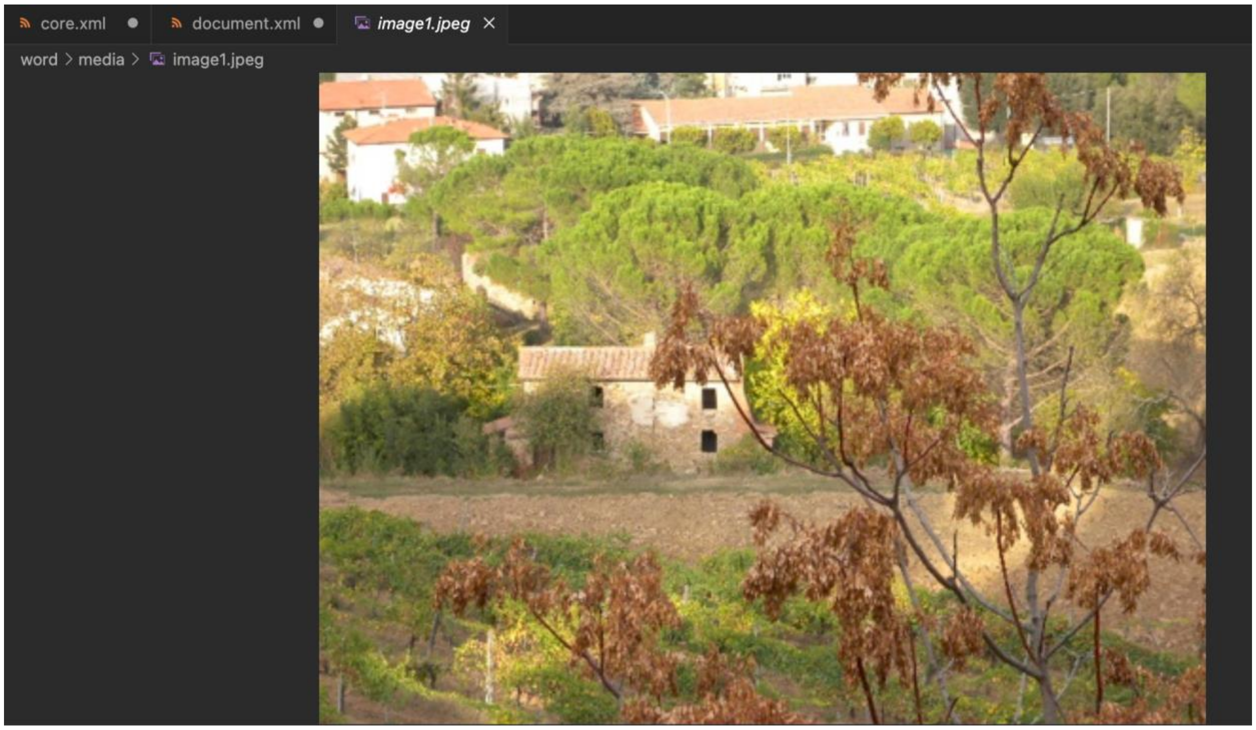


Figure 38. image1.jpeg

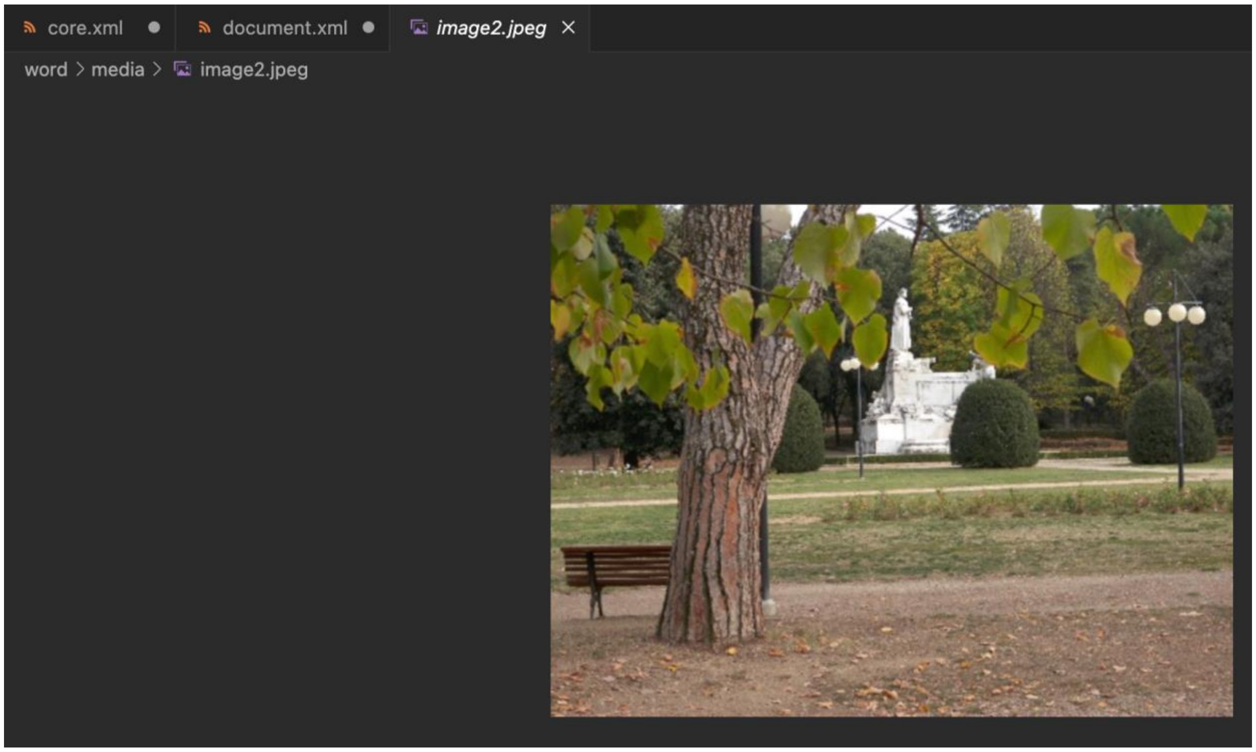


Figure 39. image2.jpeg

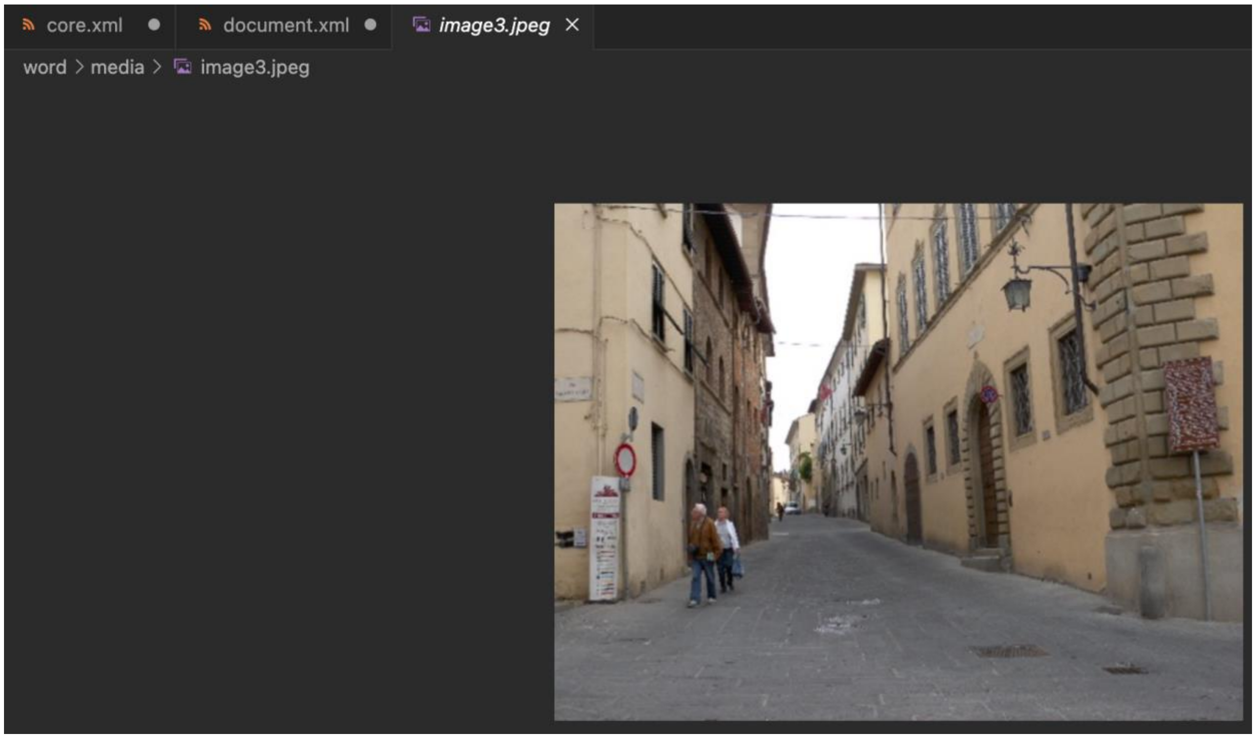


Figure 40. image3.jpeg

Conclusion

The examiner, Inor Wang, enjoyed this lab! There is no critique from me. Thank you.

References

Carvey, H. A. (2014). Windows forensic analysis toolkit: Advanced analysis techniques for Windows 8 (Fourth edition). Syngress.

Johansen, G., & Safari, an O. M. C. (2020). Digital Forensics and Incident Response—Second Edition.

Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). The art of memory forensics: Detecting malware and threats in Windows, Linux, and Mac memory. Wiley.

Malware forensics field guide for Windows systems digital forensics field guides. (2012). Syngress.

Oettinger, W., & Safari, an O. M. C. (2020). Learn Computer Forensics.