Lab Report

Name:	Inor Wang
Title:	Forensic Countermeasures
Case:	25-T111
Date:	12/05/2025

Table of Contents

Document Revision History

Name	Revision Date	Version	Description
Inor Wang	12/05/2025	0.1	Draft

Executive Summary

In this lab, the examiner used Autopsy, FTK Imager, CyberChef, Hashcat, Dominic Breuker's Stego-Toolkit, foremost, Hide'N'Seek, and supporting tools to perform static analysis of the USB image AntiForensics_A.001. The objective was to identify and recover evidence deliberately concealed with substitution ciphers, XOR encryption, steganography, file carving, misleading file extensions, embedded content in Office Open XML containers, password-protected Office documents, and NTFS metadata ($MFT). Across multiple text, document, spreadsheet, and image files, the examiner decoded or extracted diary-style narratives, detailed biographical information about the suspect, photographs of the suspect and her dog, and explicit "EVIDENCE" markers. The artifacts show a consistent pattern of intentional obfuscation designed to test the examiner's ability to recognize encoding schemes, leverage external tools, interpret file headers, and inspect low-level filesystem structures. Overall, the AntiForensics_A USB was configured as a comprehensive anti-forensics challenge, and 13 hidden EVIDENCE or suspect pictures were successfully located and documented.

Key findings:

- **Rotational cipher diary** – Information1.txt contained a diary entry encoded with a rotational cipher; ROT17 decoding in CyberChef revealed detailed biographical information about the suspect and explicitly labeled the decoded message as EVIDENCE.
- **XOR-encoded diary** – Information2.docx held another diary entry encrypted with XOR; CyberChef's XOR brute force and key 0x87 exposed additional biographical details (including the suspect's prior degree, interests, and dog's name) and identified the decoded content as EVIDENCE.
- **CipherExamples.docx image** – CipherExamples.docx explained basic ciphers and, at the end of the document, included a hidden image containing the word "EVIDENCE," confirming that this document was itself part of the anti-forensics challenge.
- **Forecast.xlsx hidden image and XML tag** – Forecast.xlsx stated it was "hiding TWO things." Inspection of its ZIP-based Office Open XML structure revealed an embedded image (image1.png) of the suspect's dog and a drawing1.xml file whose markup contained the word "EVIDENCE," providing both graphic and textual hidden artifacts.

- **Disguised Excel workbook** – Difference.docx listed Microsoft Excel as its program name; renaming it to Difference.xlsx and opening it as a spreadsheet revealed clearly marked EVIDENCE, showing that the file extension had been intentionally altered to conceal the workbook.
- **Stego in bird.jpg** – bird.jpg, analyzed with Dominic Breuker's Stego-Toolkit, contained trailing data carved into a separate file that produced a hidden image of the suspect, confirming steganographic manipulation of the original JPEG.
- **File-carved image from Ideas.docx** – Ideas.docx could not be opened in Word; file carving with foremost recovered embedded objects, including a JPEG image of the suspect, demonstrating that the document had been used as a container for hidden graphics.
- **bikes.jpg multi-layer disguise** – Hex inspection showed bikes.jpg had a BMP ("BM") header, while the file command identified it as a ZIP archive; after renaming and extracting, the recovered folder structure matched an Excel workbook whose drawings directory contained additional EVIDENCE, revealing chained header spoofing and OOXML embedding.
- **Hide'N'Seek stego on sailboat.jpg** – ROT13-decoded UserAssist artifacts showed use of the Hide'N'Seek steganography tool; running the same utility against sailboat.jpg and guessing the password "123456" produced SteggedEvidence.txt, which contained another hidden evidentiary message.
- **Password-protected Secret.docx** – office2hashcat.py and Hashcat (mode 9500, MS Office 2010) with a custom biographical dictionary in a combinator attack recovered the password "mazda_amherst." Opening Secret.docx revealed the final EVIDENCE stating that two different files had to be deciphered to build the biographical dictionary.
- **Password-protected Notes.docx** – The same workflow against Notes.docx produced the password "quinonesutsa"; opening the document displayed a confirmation that the password had been cracked and explicitly identified the document itself as EVIDENCE.
- **NTFS $MFT hidden message** – Review of the NTFS Master File Table ($MFT) in FTK Imager disclosed a human-readable string in an entry for a deleted file stating that "E-V-I-D-E-N-C-E" had been found in that MFT entry, proving that evidence had also been deliberately embedded in filesystem metadata.

Synopsis

The examiner was tasked with conducting a manual forensic examination of a 512 MB USB image to identify all instances of evidentiary material related to the suspect, Lily Quinones, a Cyber Security major at UTSA. Evidence was defined as any file or artifact containing the term “EVIDENCE” or any image of the suspect’s dog, with the additional expectation that multiple anti-forensics or data obfuscation techniques (approximately 13) would be present and needed to be recognized and addressed. The client explicitly required a technique-driven approach rather than reliance on full-featured forensic suites such as FTK or Autopsy; however, limited tools like FTK Imager were permitted to view the file system. The examiner was further instructed to analyze a provided UserAssist registry key to identify programs used by the suspect, to use open-source tools cautiously, and to ensure that any protected content was accessed using either guessable passwords or quickly crackable dictionary-based attacks. All findings were to be documented in a standard forensic report format, including metadata, methodology, tool usage, and screenshots, with the analytical strategy organized around the specific anti-forensics techniques encountered.

Client Instructions:

1. Obtain one of the following 512MB USB image files (will be assigned):
2. Find all the evidence you can.
    a. Evidence is anything containing the word ‘EVIDENCE’ or anything containing a picture of your suspect’s dog. Information about your suspect is listed below.
    b. There are approximately 13 instances of anti-forensics / data obfuscation techniques (depending on how you count an instance).
    c. You may need to apply skills and knowledge learned in Digital Forensic Analysis I.
3. Report on all evidentiary or suspicious findings.
    a. Standard forensic reporting – metadata, discussion of findings, etc.
    b. Include screenshots of your findings, including tool reports, if available (e.g. for PRTK password cracking report, and any other tools you use that has a report or log function).
    c. Include a brief overview of your analytical strategy, steps taken, tools used, etc. Organize this section of your report by anti-forensics technique.

Biographical and Case Information:

- What you initially know about your suspect: Her name is Lily Quinones. She is a Cyber Security major at the University of Texas at San Antonio (UTSA). She is currently a Senior in the College of Business. This is all you know at this point. Perhaps the files on the USB image contain more information...

- Analyze the UserAssist Registry Key provided to discover traces of programs used by the suspect.

- Do not use FTK or Autopsy (or any other similarly designed / featured "all-in-one" digital forensics tool) to complete the lab. Such tools tend to do a good job at automatically extracting and alerting you to some of the anti-forensics techniques applied here. The point of this assignment is for you to think through anti-forensic techniques and then intelligently look for traces of them; the point is not to have your tool do all the work for you. You may, however, use such a tool to check your work after you're done, or find remaining things after you've put forth all the effort you can/want to put into this lab. Write your findings up before doing so, however.

- You can (and are recommended to) use FTK Imager. You can download it free from accessdata.com. This will enable you to see the file system.

- If you use WinHex, do not use its "File Recovery by Type" feature.

- You will be graded primarily on your investigative approach and application of knowledge pertaining to anti-forensics techniques, more so than the degree of actual evidentiary discovery.

- Be careful downloading open source tools to aid you in your search. Follow lab procedures regarding software installation. Be sure not to infect your systems or the lab systems with malware when searching for tools.

- Anything that requires a password is either guessable (as it's a commonly used password), or quickly (<5 minutes) crackable with a properly selected and configured forensic dictionary. (So no lab machines should be set to run overnight trying to crack passwords/encryption in this lab assignment.)

- Every entry in your biographical dictionary should be a single lower-case word. For example, Cyber Security would be entered as "cyber" and "security".

Evidence Analyzed

This section provides details of the digital evidence collected

Evidence ID	**E001**
Name	AntiForensics_A.001
Type	DOS/MBR boot sector
Size	512 MB
MD5	14EA9F129B75747D8319118B123847AE
SHA1	1B50931A0695D8E525D61C7DEBB4690B71B540EB
SHA256	EC1F18A5B5BDD18CE438DEAB4D6C2B236B424A76B753E010E2F661DF48706936

Tools Used

Workstation

Hostname	Operating System	Build	Physical / Virtual	Built
IS-4523-001-WINDOWS	Windows 11	2021	Virtual	09/06/2025
IS-4523-001-GREYMHATTER	Fedora	2025	Virtual	10/31/2025
Inor	Windows 11	2023	Physical	02/15/2015

Software

Name	Version	Release	Purpose
FTK Imager (Access Data)	4.7.3.81	Dec 2024	A forensic imaging and preview tool that lets the examiner mount disk images, browse the file system, view hex, and export files without altering the original evidence.
CyberChef (Crown)	10.19.4	Jun 2025	A web-based “data Swiss army knife” that allows the examiner to decode, encode, and transform data (e.g., ROT ciphers, XOR, base64) through drag-and-drop operations.
Stego-Toolkit (Dominic Breuker)	4.7.3.81	Dec 2024	A Docker-based collection of steganography and file-inspection tools that automates checks on images (e.g., JPGs) for hidden content using multiple underlying utilities.
foremost	N/A	N/A	A file-carving tool that scans raw data for known file signatures and reconstructs embedded or deleted files (such as JPGs or ZIPs) from within larger containers.
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.
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).

VSCode (Microsoft)	1.94.6 (Stable)	Oct 2024	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.
Hide'N'Send	1.0.2.0	Apr 2014	A steganography application used to hide and extract data inside image files (such as JPGs), requiring the correct password to reveal the concealed content.

Analysis Findings

Overview of Examination Procedures

For this examination, the examiner began by mounting the USB image AntiForensics_A.001 into FTK Imager and conducting a systematic review of the logical file structure and NTFS metadata, including special system files such as $MFT. Suspect files were exported as needed for deeper analysis, including text documents, Office documents, and multiple image files. The examiner used CyberChef to decode obfuscated content (ROT-based ciphers and XOR encryption) recovered from *Information1.txt* and *Information2.docx*, and relied on office2hashcat.py with Hashcat to extract and crack password hashes for protected Word documents (*Secret.docx* and *Notes.docx*) using a custom biographical wordlist. Potential steganographic images (*bird.jpg*, *sailboat.jpg*, and others) were examined inside a controlled Docker container running Dominic Breuker's Stego-Toolkit, while corrupted or unreadable documents (such as *Ideas.docx*) were subjected to file carving with foremost to recover embedded objects. The examiner also identified files with misleading extensions (e.g., Excel workbooks disguised as .docx or .jpg) by inspecting hex signatures and using the file command, then renamed and extracted the underlying ZIP-based Office Open XML structures to locate hidden drawings and images. Finally, registry-based artifacts (UserAssist) and NTFS metadata **(**$MFT) were inspected within FTK Imager's hex/ASCII viewer to identify usage of anti-forensic tools and to locate messages intentionally hidden in deleted-file entries, ensuring that all potential locations for concealed EVIDENCE on the USB image were thoroughly examined. Additional targeted analysis was performed using:

- **Autopsy (Basis Technology)** — Used to analyze the disk image file and review files/artifacts at a high level.
- **FTK Imager (Exterro FTK Imager 4.7.3.81)** — Used to mount the AntiForensics_A.001 image, browse the NTFS file system, examine hex/ASCII views, and inspect metadata files such as $MFT.
- **CyberChef** — Web-based decoding tool used to perform ROT13/ROT17 and XOR operations, as well as decode the UserAssist data.
- **Hashcat** — GPU-accelerated password-cracking tool used to attack MS Office 2010 hashes for Secret.docx and Notes.docx with custom dictionary and combinator attacks.
- **office2hashcat.py** — Python script used to extract and convert protected Office document passwords into Hashcat-compatible hash formats.

- **Dominic Breuker's Stego-Toolkit (Docker image)** — Steganography toolkit used (via check_jpg.sh) to analyze bird.jpg and other JPEGs for hidden content and trailing data.
- **foremost** — File-carving utility used to recover embedded objects (e.g., JPG and ZIP files) from the corrupt Ideas.docx document.
- **Hide'N'Seek** — Steganography program used to extract the hidden file SteggedEvidence.txt from sailboat.jpg after supplying the correct password.

Throughout the process, all findings were documented and evidence files were correctly hashed.

Evidence Reviewed

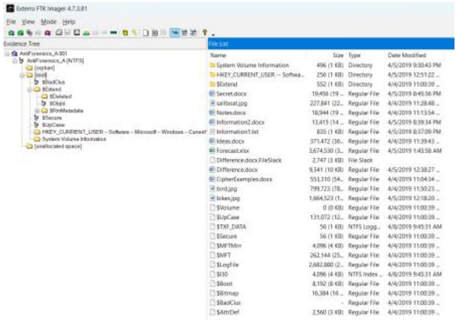
AntiForensics_A.001(E01): USB image A

Key Findings

1. Information1.txt

- **Analysis Performed:**
  - The examiner mounted the file (USB image A, "AntiForensics_A.001") into FTK Imager and went through the files as shown in Figure 1.
  - The examiner then went to "Information1.txt" and noticed that there seemed to be an encoded message with a rotational cipher as shown in Figure 2.
  - The examiner then proceeded to CyberChef, an open-source and web-based tool often used for decoding/encoding and encrypting/decrypting, and pasted the message into the input field. Then, the examiner applied a ROT13 decode operation and changed the alphabet position shift by 17 to properly decode the message as shown in Figure 3.
- **Answer:**
  The examiner used CyberChef with a ROT17 decode operation. Due to the operations completed in Figures 1, 2, and 3, the decoded message of "Information1.txt" is: "**You open a text file on the suspect's USB image and find that it's been encoded with a rotational cipher. This doesn't stop you! You decode the message and find that it is a diary entry that your suspect has written which gives you tons of biographical information. You learn that your subject has taken the following courses: Information Assurance and Security, Network Security, Digital Forensic Analysis, Operating Systems Security, and Intrusion Detection and Incident Response. Her favorite subject is forensics which is taught by Dr. Nicole Beebe. She likes programming in Java and is currently learning Python. She speaks Spanish and drives a Mazda. This information could prove highly useful! Maybe there's more biographical information hiding on the USB image...Either way, the message you have decoded is considered EVIDENCE!**".

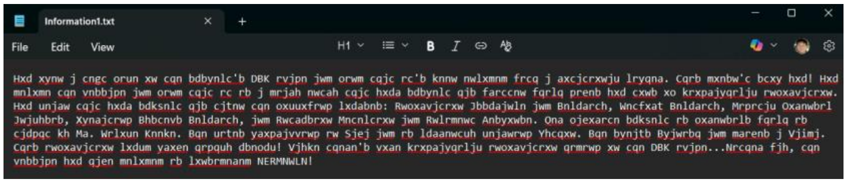
- **Supporting Evidence:**



AntiForensics_A.001
AntiForensics_A [NTFS]
[orphan]
[root]
$BadClus
$Extend
$Deleted
$ObjId
$RmMetadata
$Secure
$UpCase
HKEY_CURRENT_USER -- Software -- Microsoft -- Windows -- CurrentV
System Volume Information
[unallocated space]

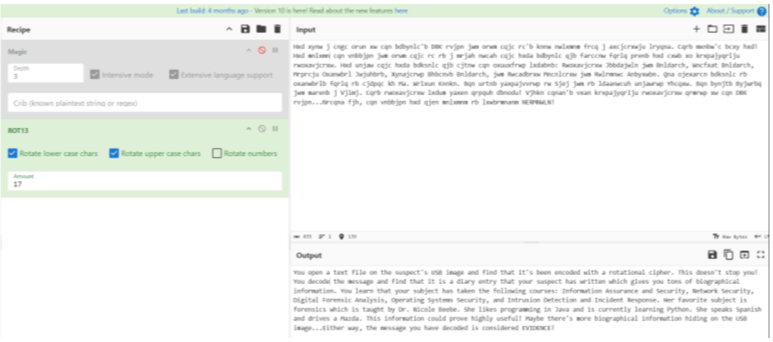
Name	Size	Type	Date Modified
System Volume Information	496 (1 KB)	Directory	4/5/2019 9:30:43 PM
HKEY_CURRENT_USER -- Softwa...	256 (1 KB)	Directory	4/5/2019 12:51:22 ...
$Extend	552 (1 KB)	Directory	4/4/2019 11:00:39 ...
Secret.docx	19,456 (19 ...	Regular File	4/5/2019 8:45:36 PM
sailboat.jpg	227,841 (22...	Regular File	4/4/2019 11:28:48 ...
Notes.docx	18,944 (19 ...	Regular File	4/4/2019 11:13:54 ...
Information2.docx	13,415 (14 ...	Regular File	4/5/2019 8:39:34 PM
Information1.txt	835 (1 KB)	Regular File	4/5/2019 8:37:09 PM
Ideas.docx	371,472 (36...	Regular File	4/4/2019 11:39:43 ...
Forecast.xlsx	3,674,530 (3...	Regular File	4/5/2019 1:45:58 AM
Difference.docx.FileSlack	2,747 (3 KB)	File Slack	
Difference.docx	9,541 (10 KB)	Regular File	4/5/2019 12:38:27 ...
CipherExamples.docx	553,310 (54...	Regular File	4/4/2019 11:04:34 ...
bird.jpg	799,723 (78...	Regular File	4/4/2019 11:50:23 ...
bikes.jpg	1,664,523 (1...	Regular File	4/5/2019 12:18:20 ...
$Volume	0 (0 KB)	Regular File	4/4/2019 11:00:39 ...
$UpCase	131,072 (12...	Regular File	4/4/2019 11:00:39 ...
$TXF_DATA	56 (1 KB)	NTFS Logg...	4/8/2019 9:45:31 AM
$Secure	56 (1 KB)	Regular File	4/4/2019 11:00:39 ...
$MFTMirr	4,096 (4 KB)	Regular File	4/4/2019 11:00:39 ...
$MFT	262,144 (25...	Regular File	4/4/2019 11:00:39 ...
$LogFile	2,682,880 (2...	Regular File	4/4/2019 11:00:39 ...
$I30	4,096 (4 KB)	NTFS Index ...	4/8/2019 9:45:31 AM
$Boot	8,192 (8 KB)	Regular File	4/4/2019 11:00:39 ...
$Bitmap	16,384 (16 ...	Regular File	4/4/2019 11:00:39 ...
$BadClus	-	Regular File	4/4/2019 11:00:39 ...
$AttrDef	2,560 (3 KB)	Regular File	4/4/2019 11:00:39 ...

Figure 1. Mounting AntiForensics_A.001 into FTK Imager



```
Hxd xynw j cngc orun xw cqn bdbynlc'b DBK rvjpn jwm orwm cqjc rc'b knnw nwlxmnm frcq j axcjcrxwju lryqna. Cqrb mxnbw'c bcxy hxd! Hxd
mnlxmn cqn vnbbjpn jwm orwm cqjc rc rb j mrjah nwcah cqjc hxda bdbynlc qjb farccnw fqrlq prenb hxd cxwb xo krxpajyqrlju rwoxavjcrxw.
Hxd unjaw cqjc hxda bdksnlc qjb cjtnw cqn oxuuxfrwp lxdabnb: Rwoxavjcrxw Jbbdajwln jwm Bnldarch, Wncfxat Bnldarch, Mrprcju Oxanwbrl
Jwjuhbrb, Xynajcrwp Bhbcnvb Bnldarch, jwm Rwcadbrxw Mncnlcrxw jwm Rwlrmnwc Anbyxwbn. Qna ojexarcn bdksnlc rb oxanwbrlb fqrlq rb
cjdpqc kh Ma. Wrlxun Knnkn. Bqn urtnb yaxpajvvrwp rw Sjej jwm rb ldaanwcuh unjawrwp Yhcqxw. Bqn bynjtb Byjwrbq jwm marenb j Vjimj.
Cqrb rwoxavjcrxw lxdum yaxen qrpquh dbnodu! Vjhkn cqnan'b vxan krxpajyqrlju rwoxavjcrxw qrmrwp xw cqn DBK rvjpn...Nrcqna fjh, cqn
vnbbjpn hxd qjen mnlxmnm rb lxwbrmnanm NERMNWLN!
```

Figure 2. Exported "Information1.txt" which shows an encrypted message.



Last build: 4 months ago - Version 10 is here! Read about the new features here
Options
About / Support
Recipe
Input
Output
835
1
139
Raw Bytes

Magic
Depth
3
☑ Intensive mode ☑ Extensive language support
Crib (known plaintext string or regex)

ROT13
☑ Rotate lower case chars ☑ Rotate upper case chars ☐ Rotate numbers
Amount
17

Input

```
Hxd xynw j cngc orun xw cqn bdbynlc'b DBK rvjpn jwm orwm cqjc rc'b knnw nwlxmnm frcq j axcjcrxwju lryqna. Cqrb mxnbw'c bcxy hxd!
Hxd mnlxmn cqn vnbbjpn jwm orwm cqjc rc rb j mrjah nwcah cqjc hxda bdbynlc qjb farccnw fqrlq prenb hxd cxwb xo krxpajyqrlju
rwoxavjcrxw. Hxd unjaw cqjc hxda bdksnlc qjb cjtnw cqn oxuuxfrwp lxdabnb: Rwoxavjcrxw Jbbdajwln jwm Bnldarch, Wncfxat Bnldarch,
Mrprcju Oxanwbrl Jwjuhbrb, Xynajcrwp Bhbcnvb Bnldarch, jwm Rwcadbrxw Mncnlcrxw jwm Rwlrmnwc Anbyxwbn. Qna ojexarcn bdksnlc rb
oxanwbrlb fqrlq rb cjdpqc kh Ma. Wrlxun Knnkn. Bqn urtnb yaxpajvvrwp rw Sjej jwm rb ldaanwcuh unjawrwp Yhcqxw. Bqn bynjtb Byjwrbq
jwm marenb j Vjimj. Cqrb rwoxavjcrxw lxdum yaxen qrpquh dbnodu! Vjhkn cqnan'b vxan krxpajyqrlju rwoxavjcrxw qrmrwp xw cqn DBK
rvjpn...Nrcqna fjh, cqn vnbbjpn hxd qjen mnlxmnm rb lxwbrmnanm NERMNWLN!
```

Output

```
You open a text file on the suspect's USB image and find that it's been encoded with a rotational cipher. This doesn't stop you!
You decode the message and find that it is a diary entry that your suspect has written which gives you tons of biographical
information. You learn that your subject has taken the following courses: Information Assurance and Security, Network Security,
Digital Forensic Analysis, Operating Systems Security, and Intrusion Detection and Incident Response. Her favorite subject is
forensics which is taught by Dr. Nicole Beebe. She likes programming in Java and is currently learning Python. She speaks Spanish
and drives a Mazda. This information could prove highly useful! Maybe there's more biographical information hiding on the USB
image...Either way, the message you have decoded is considered EVIDENCE!
```

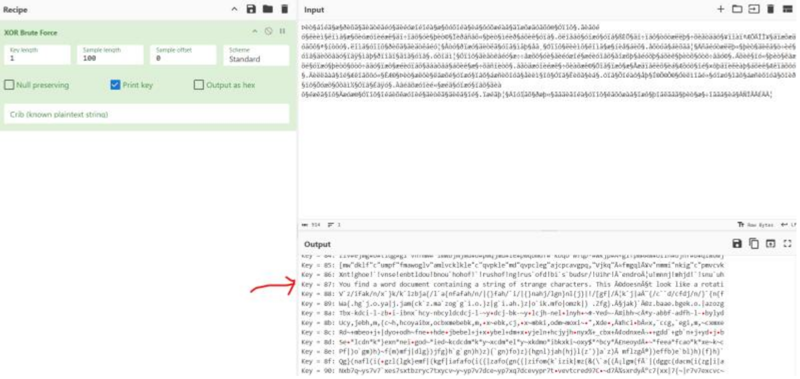
Figure 3. Decoded the rotational encoded message in "Information1.txt"

2. Information2.docx

- **Analysis Performed:**
  - The examiner then went to "Information2.docx" and noticed that there seemed to be an encoded message with XOR as shown in Figure 4.
  - The examiner then proceeded to CyberChef, an open-source and web-based tool often used for decoding/encoding and encrypting/decrypting, and pasted the message into the input field. Then, the examiner applied a XOR Brute Force decode operation to find the correct Hex key as shown in Figure 5.
  - The examiner then proceeded to change the decode operation to XOR with a hex key of 87, as shown in Figure 6.
- **Answer:**
  The examiner used a XOR decode operation with a key of 87. Due to the operations completed in Figures 4, 5, and 6, the decoded message of "Information2.docx" is: "**You find a word document containing a string of strange characters. This ©doesn§t look like a rotational cipher to you. However, you know from the ©slides that the XOR cipher usually produces "high-ASCII" characters - just ©like this word document! But what could the key be? This looks like a job for ©brute force! Eventually, you come upon the correct hex key which did the ©trick! This document appears to contain another diary entry from your suspect. ©From it, you learn that your suspect has another degree from a previous ©educational program. She has a Bachelor of Arts in Psychology from Amherst ©College in Amherst, MA. You also learn that her favorite book is The Lord of ©the Rings by J.R.R. Tolkien, that her favorite show is Star Trek: The Next ©Generation, and that her dog§s name is Data. This information could come in ©handy! Either way, decoding this message has yielded you a piece of EVIDENCE!**".

- **Supporting Evidence:**

Figure 4. Exported "Information2.docx" which shows an encrypted message.



Recipe

XOR Brute Force

Key length: 1

Sample length: 100

Sample offset: 0

Scheme: Standard

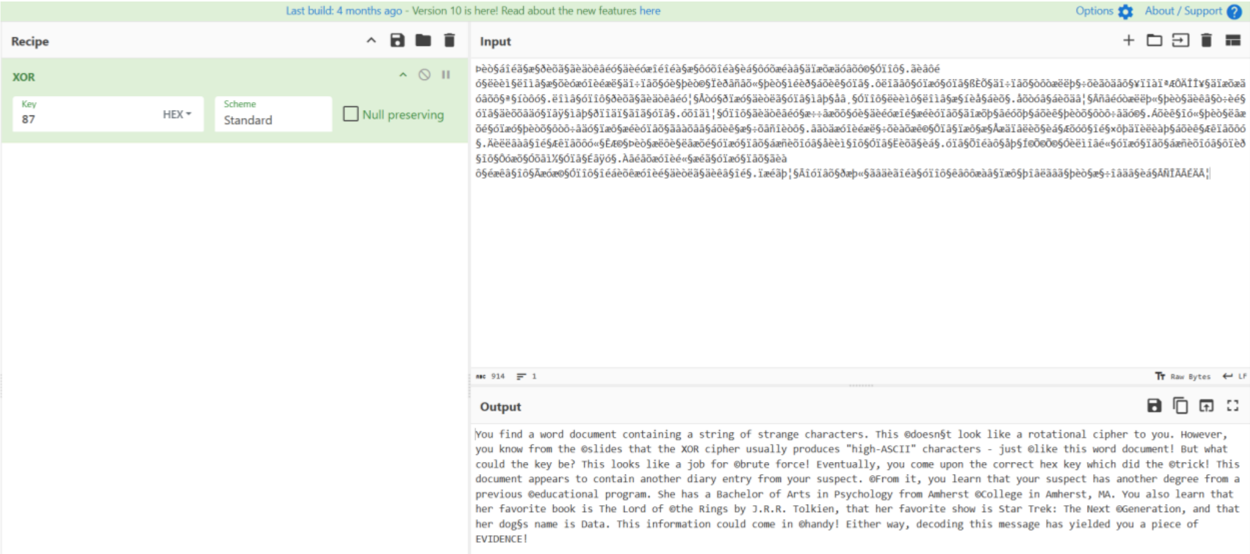
□ Null preserving ☑ Print key □ Output as hex

Crib (known plaintext string)

Output

```
Key = 85: [mw"dklf"c"umpf"fmawoglv"amlvcklkle"c"qvpkle"md"qvpcleg"ajcpcavgpq,"Vjkq"Â«fmgqlÂ¥v"nmmi"nkig"c"pmvcvk
Key = 86: Xnt!ghoe!`!vnse!enbtldou!bnou`hohof!`!rushof!ng!rus`ofd!bi`s`budsr/!Uihr!Â`endroÂ¦u!mnnj!mhjd!`!snu`uh
Key = 87: You find a word document containing a string of strange characters. This Â©doesnÂ§t look like a rotati
Key = 88: V`z/ifak/n/x`}k/k`lzbja{/l`a{nfafah/n/|{}fah/`i/|{}nahj/lgn}nl{j}|!/[gf|/Â¦k`j|aÂ`{/c``d/cfdj/n/}`{n{f
Key = 89: Wa{.hg`j.o.ya|j.jam{ck`z.ma`zog`g`i.o.}z|g`i.ah.}z|o`ik.mfo|omzk|} .Zfg}.Â§jak}`Â®z.baao.bgek.o.|azozg
Key = 8a: Tbx-kdci-l-zb-i-ibnx`hcy-nbcyldcdcj-l-~y-dcj-bk-~y-lcjh-nel-lnyh-~-Yed~-Â¤ibh~<Â*y-abbf-adfh-l--bylyd
Key = 8b: Ucy,jebh,m,{c~h,hcoyaibx,ocbxmebebk,m,-x~ebk,cj,-x~mbkl,odm~moxl~-",Xde-,Â¤hcl-bÂ«x,`ccg,`egl,m,~cxmxe
Key = 8c: Rd~+mbeo+j+|dyo+odh~fne++hde-jbebel+j+x-ybel+dm+x-yjeln+hcjyjh-nyx%+_cbx+Â¢odnxeÂ-+gdd`+gb`n+j+yd-j-b
Key = 8d: Se-*lcdn*k*}exn*nei-god~*ied~kcdcdm*k*y~xcdm*el*y~xkdmo*ibkxkl~oxy$*^bcy*Â£neoydÂ-~*feea*fcao*k*xe~k~c
Key = 8e: Pf|)o`gm)h)~f{m)mfj|dlg})jfg}h`g`gn)h)z}(`gn)fo)z}{hgnl}jah{hj}l(z`)}a`z)Â mflzgÂ®})effb)e`bl)h)(f}h)`
Key = 8f: Qg}(nafl(i(+gzl(lgk}emf|(kgf|iafafo(i({|zafo(gn({|zifom(k`izik|mz{&(\`a{(Â¡lgm{fÂ`|(dggc(dacm(i(zg|i|a
Key = 90: Nxb7q~ys7v7`xes7sxtbzryc7txycv~y~yp7v7dce~yp7xq7dcevypr7t-vevtcred97C-~d7Â¾sxrdyÂ°c7{xx|7{~|r7v7excvc~
```

Figure 5. Utilized XOR Brute Force to find the key


Last build: 4 months ago - Version 10 is here! Read about the new features here
Options
About / Support
Recipe
Input
Output
914
1
Raw Bytes
LF

XOR

Key: 87 HEX

Scheme: Standard

☐ Null preserving

Input

```
Þèò§áîéã§æ§ðèõã§ãèäòêâéó§äèéóæîéîéà§æ§ôóõîéà§èá§ôóõæéàâ§äïæõæäóâõô©§Óïîô§.ãèâôé
ó§ëèèì§ëîìâ§æ§õèóæóîèéæë§äî÷ïâõ§óè§þèò©§Ïèðâñâõ«§þèò§ìéèð§áõèê§óïâ§.ôëîãâô§óïæó§óïâ§ßÈÕ§äî÷ïâõ§òôòæëëþ§÷õèãòäâô§¥ïîàïªÆÔÄÎÎ¥§äïæõæä
óâõô§ª§íòôó§.ëîìâ§óïîô§ðèõã§ãèäòêâéó¦§Åòó§ðïæó§äèòëã§óïâ§ìâþ§åâ¸§Óïîô§ëèèìô§ëîìâ§æ§íèå§áèõ§.åõòóâ§áèõäâ¦§Âñâéóòæëëþ«§þèò§äèêâ§ò÷èé§
óïâ§äèõõâäó§ïâÿ§ìâþ§ðïîäï§ãîã§óïâ§.óõîäì¦§Óïîô§ãèäòêâéó§æ÷÷âæõô§óè§äèéóæîé§æéèóïâõ§ãîæõþ§âéóõþ§áõèê§þèòõ§ôòô÷âäó©§.Áõèê§îó«§þèò§ëâæ
õé§óïæó§þèòõ§ôòô÷âäó§ïæô§æéèóïâõ§ãâàõââ§áõèê§æ§÷õâñîèòô§.âãòäæóîèéæë§÷õèàõæê©§Ôïâ§ïæô§æ§Åæäïâëèõ§èá§Æõóô§îé§×ôþäïèëèàþ§áõèê§Æêïâõôó
§.Äèëëâàâ§îé§Æêïâõôó«§ÊÆ©§Þèò§æëôè§ëâæõé§óïæó§ïâõ§áæñèõîóâ§åèèì§îô§Óïâ§Ëèõã§èá§.óïâ§Õîéàô§åþ§Í©Õ©Õ©§Óèëìîâé«§óïæó§ïâõ§áæñèõîóâ§ôïèð
§îô§Ôóæõ§Óõâì½§Óïâ§Éâÿó§.Àâéâõæóîèé«§æéã§óïæó§ïâõ§ãèà
ô§éæêâ§îô§Ãæóæ©§Óïîô§îéáèõêæóîèé§äèòëã§äèêâ§îé§.ïæéãþ¦§Âîóïâõ§ðæþ«§ãâäèãîéà§óïîô§êâôôæàâ§ïæô§þîâëãâã§þèò§æ§÷îâäâ§èá§ÂÑÎÃÂÉÄÂ¦
```

Output

```
You find a word document containing a string of strange characters. This ©doesn§t look like a rotational cipher to you. However,
you know from the ©slides that the XOR cipher usually produces "high-ASCII" characters - just ©like this word document! But what
could the key be? This looks like a job for ©brute force! Eventually, you come upon the correct hex key which did the ©trick! This
document appears to contain another diary entry from your suspect. ©From it, you learn that your suspect has another degree from a
previous ©educational program. She has a Bachelor of Arts in Psychology from Amherst ©College in Amherst, MA. You also learn that
her favorite book is The Lord of ©the Rings by J.R.R. Tolkien, that her favorite show is Star Trek: The Next ©Generation, and that
her dog§s name is Data. This information could come in ©handy! Either way, decoding this message has yielded you a piece of
EVIDENCE!
```

Figure 6. Correctly utilized the key to decode the XOR message

3. CipherExamples.docx

- **Analysis Performed:**
  - The examiner then went to examine "CipherExamples.docx", the document contained a lot of information about encoding/decoding, more specifically about ROT13 which provided useful for decoding the message within "Information1.txt", however as scrolling down, the examiner noted an image that states "EVIDENCE", as shown in Figure 7.
- **Answer:**

  After reading the full content of "CipherExamples.docx", the examiner noted that an image of the word "EVIDENCE" was hidden at the very bottom of the document, as shown in Figure 7.

- **Supporting Evidence:**

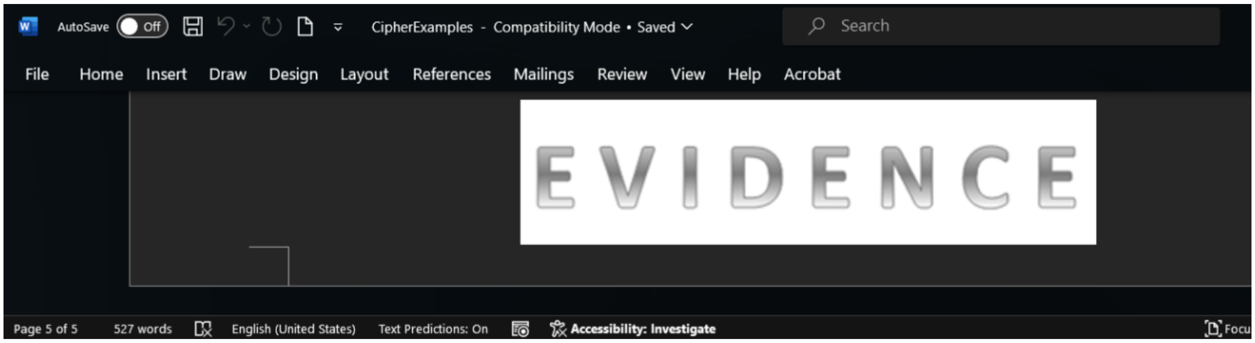
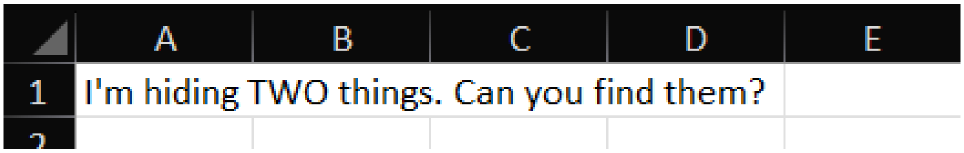


Figure 7. "CipherExamples.docx" containing the word, "EVIDENCE"

4. Forecast.xlsx

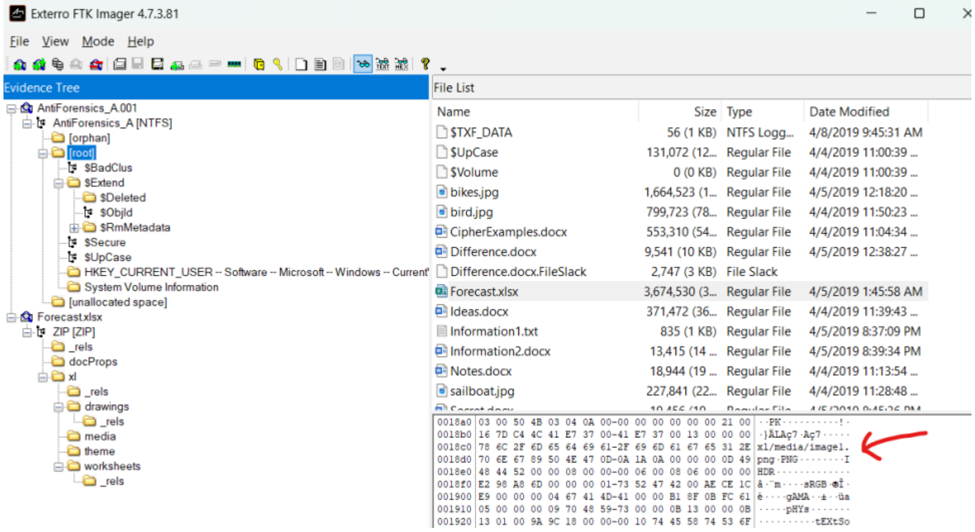
- **Analysis Performed:**
  - The examiner then proceeded to “Forecast.xlsx”, the examiner exported and opened the Microsoft Excel file and noticed that it states, “I'm hiding TWO things. Can you find them?”, as shown in Figure 8.
  - The examiner then went to examine the hex portion of the file and noticed that there was a image1.png embedded into the file as shown in Figure 9. Excel stores .xlsx files as ZIP archives which contains multiple XML files and metadata folders which means that an image was hidden into the ZIP archive.
  - The examiner then mounted “Forecast.xlsx” into FTK Imager to find the ZIP archive which contained an image of the suspect (the dog), as shown in Figure 10.
  - The examiner then turned the excel file into a ZIP archive, extracted, and mounted it into Visual Studio Code. Then the examiner went through the file system, noting that the image1.png is there as mentioned before, but also noticed a file named, “drawing1.xml” which contains text, “EVIDENCE”, as shown in Figure 11.
- **Answer:**
  By reviewing Forecast.xlsx, the examiner confirmed that the workbook’s banner text—“I’m hiding TWO things. Can you find them?”—was accurate. Inspection of the file’s hex data and internal ZIP structure showed that the spreadsheet was really an Office Open XML container with an embedded image file, image1.png, which, when extracted, revealed a picture of the suspect (the dog). Further review of the extracted contents identified drawing1.xml, whose XML markup included the word “EVIDENCE”, providing a second, text-based hidden item within the same workbook. Together, these artifacts demonstrate that Forecast.xlsx was deliberately used to conceal both a covert image of the suspect and a textual EVIDENCE marker. The examiner found image1.png in the ZIP archive within “Forecast.xlsx” which contained an image of the suspect, as shown in Figure 10. The examiner also went to a file named, “drawing1.xml”, within the ZIP archive that contained the text, “EVIDENCE”, as shown in Figure 11.

- **Supporting Evidence:**



I'm hiding TWO things. Can you find them?

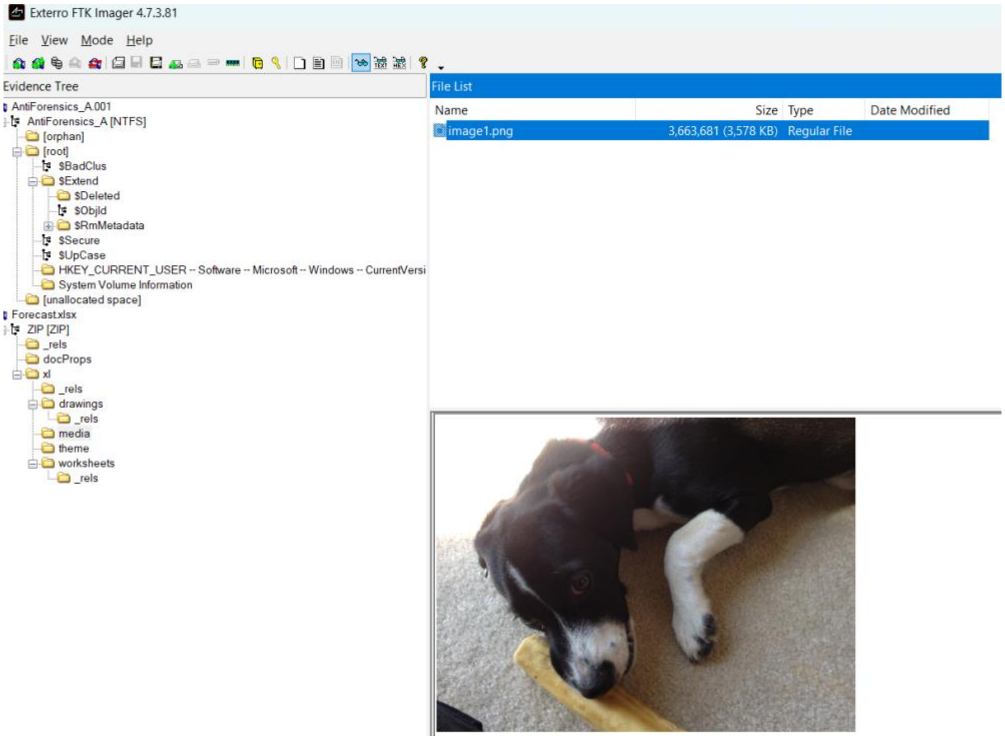
Figure 8. Accessed the "Forecast.xlsx" file



Name	Size	Type	Date Modified
$TXF_DATA	56 (1 KB)	NTFS Logg...	4/8/2019 9:45:31 AM
$UpCase	131,072 (12...	Regular File	4/4/2019 11:00:39 ...
$Volume	0 (0 KB)	Regular File	4/4/2019 11:00:39 ...
bikes.jpg	1,664,523 (1...	Regular File	4/5/2019 12:18:20 ...
bird.jpg	799,723 (78...	Regular File	4/4/2019 11:50:23 ...
CipherExamples.docx	553,310 (54...	Regular File	4/4/2019 11:04:34 ...
Difference.docx	9,541 (10 KB)	Regular File	4/5/2019 12:38:27 ...
Difference.docx.FileSlack	2,747 (3 KB)	File Slack	
Forecast.xlsx	3,674,530 (3...	Regular File	4/5/2019 1:45:58 AM
Ideas.docx	371,472 (36...	Regular File	4/4/2019 11:39:43 ...
Information1.txt	835 (1 KB)	Regular File	4/5/2019 8:37:09 PM
Information2.docx	13,415 (14 ...	Regular File	4/5/2019 8:39:34 PM
Notes.docx	18,944 (19 ...	Regular File	4/4/2019 11:13:54 ...
sailboat.jpg	227,841 (22...	Regular File	4/4/2019 11:28:48 ...

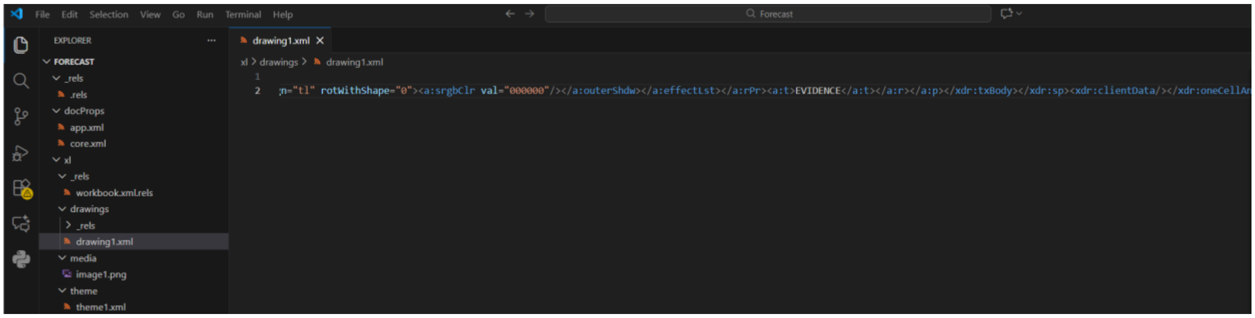
```
0018a0 03 00 50 4B 03 04 0A 00-00 00 00 00 00 00 21 00  ··PK··········!·
0018b0 16 7D C4 4C 41 E7 37 00-41 E7 37 00 13 00 00 00  ·}ÄLAç7·Aç7·····
0018c0 78 6C 2F 6D 65 64 69 61-2F 69 6D 61 67 65 31 2E  xl/media/image1.
0018d0 70 6E 67 89 50 4E 47 0D-0A 1A 0A 00 00 00 0D 49  png·PNG········I
0018e0 48 44 52 00 00 08 00 00-00 06 00 08 06 00 00 00  HDR·············
0018f0 E2 98 A8 6D 00 00 00 01-73 52 47 42 00 AE CE 1C  â·¨m····sRGB·®Î·
001900 E9 00 00 00 04 67 41 4D-41 00 00 B1 8F 0B FC 61  é····gAMA··±··üa
001910 05 00 00 00 09 70 48 59-73 00 00 0B 13 00 00 0B  ·····pHYs·······
001920 13 01 00 9A 9C 18 00 00-00 10 74 45 58 74 53 6F  ··········tEXtSo
```

Figure 9. Hex showing an image1.png file in the ZIP archive of the .xlsx file



Name	Size	Type	Date Modified
image1.png	3,663,681 (3,578 KB)	Regular File	

Figure 10. Mounted "Forecast.xlsx" to find image1.png which is a picture of the suspect



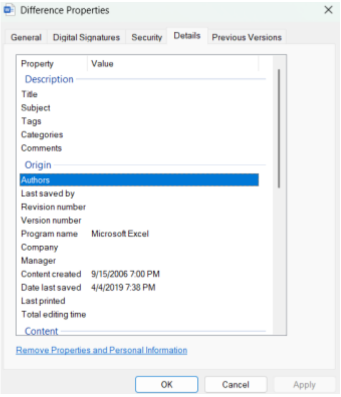
```
n="tl" rotWithShape="0"><a:srgbClr val="000000"/></a:outerShdw></a:effectLst></a:rPr><a:t>EVIDENCE</a:t></a:r></a:p></xdr:txBody></xdr:sp><xdr:clientData/></xdr:oneCellAn
```

Figure 11. "drawing1.xml" containing the text, "EVIDENCE"

5. Difference.docx

- **Analysis Performed:**
    - The examiner looked at the properties of “Difference.docx” and noted that the Program name of the file is “Microsoft Excel”, as shown in Figure 12.
    - The examiner then changed the file extension of “Difference.docx” to “Difference.xlsx”, as shown in Figure 13.
    - The examiner then accessed “Difference.xlsx” which contained EVIDENCE, as shown in Figure 14.
- **Answer:**
  By reviewing the file properties for Difference.docx, the examiner observed that the program name was Microsoft Excel, indicating that the document was in fact an Excel workbook disguised with a Word (.docx) extension. After renaming the file to Difference.xlsx and opening it as a spreadsheet, the examiner confirmed that the workbook contained clearly marked EVIDENCE, showing that the file extension had been intentionally manipulated to conceal this information, as shown in Figure 14.

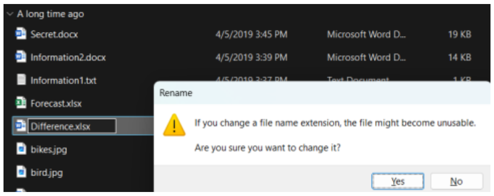
- **Supporting Evidence:**



Difference Properties

Property	Value
Description	
Title	
Subject	
Tags	
Categories	
Comments	
Origin	
Authors	
Last saved by	
Revision number	
Version number	
Program name	Microsoft Excel
Company	
Manager	
Content created	9/15/2006 7:00 PM
Date last saved	4/4/2019 7:38 PM
Last printed	
Total editing time	
Content	

Figure 12. File properties of "Difference.docx"



A long time ago

Secret.docx	4/5/2019 3:45 PM	Microsoft Word D...	19 KB
Information2.docx	4/5/2019 3:39 PM	Microsoft Word D...	14 KB
Information1.txt	4/5/2019 3:37 PM	Text Document	1 KB
Forecast.xlsx			
Difference.xlsx			
bikes.jpg			
bird.jpg			

Rename

If you change a file name extension, the file might become unusable.

Are you sure you want to change it?

Figure 13. Changing the file extension of Difference.docx to Difference.xlsx

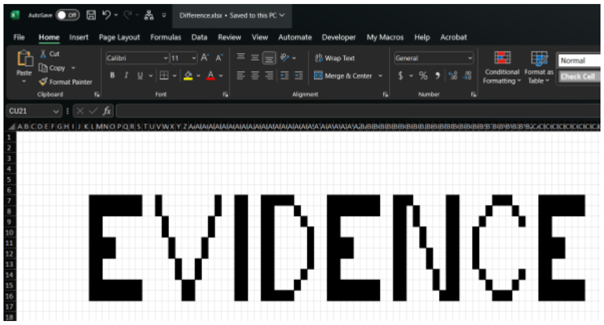
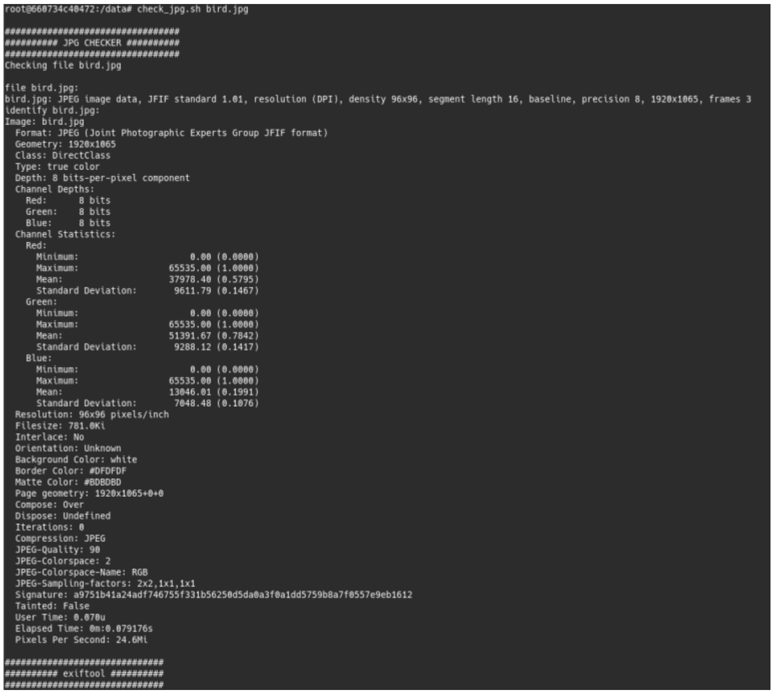


Figure 14. Accessing "Difference.xlsx"

6. bird.jpg

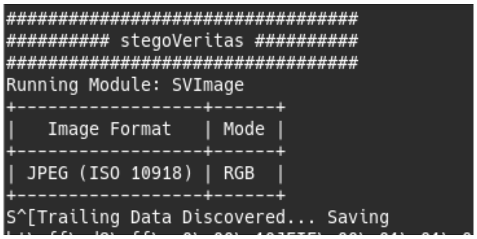
- **Analysis Performed:**
  - The examiner then noticed that there were multiple .jpg files which could possible contain steganography. The examiner decided to use Dominic Breuker's steganography tool kit. The examiner did the following commands in the terminal within Greymhatter VM machine.
    - cd Desktop
    - docker pull dominicbreuker/stego-toolkit
      - Downloaded the Stego-Toolkit Docker image from Docker Hub
    - git clone https://github.com/DominicBreuker/stego-toolkit
      - Copied the Stego-Toolkit source code repo to the machine
    - cd stego-toolkit
    - docker build -t stego .
      - Built a custom Docker image named "stego" from the toolkit's Dockerfile
    - docker run -it --rm -v /Desktop/Evidence/T111:/data dominicbreuker/stego-toolkit /bin/bash
      - Started the stego-toolkit container and mounted the evidence folder into /data for analysis
  - The examiner then used the command, "check_jpg.sh bird.jpg", to check the file for any steganography as shown in Figure 15. This script includes exiftool, binwalk, stegdetect, strings, steghide, outguess, outguess-0.13, jsteg, and stegoVeritas.
  - The command outputted a folder that contains multiple images and also any trailing data that is discovered, which from the terminal, it states that trailing data is discovered, and it is saving as shown in Figure 16.
  - Within the directory as shown in Figure 17, it contains an image of the suspect as shown in Figure 18.
- **Answer:**
  The steganography analysis of bird.jpg revealed that the image had been deliberately used to conceal additional content. After identifying multiple JPEG files of interest, the examiner loaded the evidence into Dominic Breuker's Stego-Toolkit and executed the check_jpg.sh bird.jpg script, which chains together several screening tools (including exiftool, binwalk, stegdetect, strings, steghide, outguess, jsteg, and stegoVeritas). The toolkit reported the presence of trailing data appended beyond the normal end of the JPEG structure and automatically carved this portion out as trailing_data.bin. When the examiner reviewed the extracted output directory, one of the recovered files was a separate image depicting the suspect. This hidden image had not been visible when viewing the original bird photograph in a standard image viewer. The successful recovery of the suspect's photograph from the carved trailing data confirms that bird.jpg had been intentionally manipulated using steganographic techniques to hide incriminating imagery, and that the embedded file directly links the suspect to the activity under investigation, as shown in Figure 18.

- **Supporting Evidence:**



```
root@660734c40472:/data# check_jpg.sh bird.jpg

################################
########## JPG CHECKER ##########
################################
Checking file bird.jpg

file bird.jpg:
bird.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 96x96, segment length 16, baseline, precision 8, 1920x1065, frames 3
identify bird.jpg:
Image: bird.jpg
  Format: JPEG (Joint Photographic Experts Group JFIF format)
  Geometry: 1920x1065
  Class: DirectClass
  Type: true color
  Depth: 8 bits-per-pixel component
  Channel Depths:
    Red:      8 bits
    Green:    8 bits
    Blue:     8 bits
  Channel Statistics:
    Red:
      Minimum:                    0.00 (0.0000)
      Maximum:                65535.00 (1.0000)
      Mean:                   37978.40 (0.5795)
      Standard Deviation:      9611.79 (0.1467)
    Green:
      Minimum:                    0.00 (0.0000)
      Maximum:                65535.00 (1.0000)
      Mean:                   51391.67 (0.7842)
      Standard Deviation:      9288.12 (0.1417)
    Blue:
      Minimum:                    0.00 (0.0000)
      Maximum:                65535.00 (1.0000)
      Mean:                   13046.01 (0.1991)
      Standard Deviation:      7048.48 (0.1076)
  Resolution: 96x96 pixels/inch
  Filesize: 781.0Ki
  Interlace: No
  Orientation: Unknown
  Background Color: white
  Border Color: #DFDFDF
  Matte Color: #BDBDBD
  Page geometry: 1920x1065+0+0
  Compose: Over
  Dispose: Undefined
  Iterations: 0
  Compression: JPEG
  JPEG-Quality: 90
  JPEG-Colorspace: 2
  JPEG-Colorspace-Name: RGB
  JPEG-Sampling-factors: 2x2,1x1,1x1
  Signature: a9751b41a24adf746755f331b56250d5da0a3f0a1dd5759b8a7f0557e9eb1612
  Tainted: False
  User Time: 0.070u
  Elapsed Time: 0m:0.079176s
  Pixels Per Second: 24.6Mi

##############################
########## exiftool ##########
##############################
```

Figure 15. check_jpg.sh script on bird.jpg



```
##################################
########## stegoVeritas ##########
##################################
Running Module: SVImage
+------------------+------+
|   Image Format   | Mode |
+------------------+------+
| JPEG (ISO 10918) | RGB  |
+------------------+------+
S^[Trailing Data Discovered... Saving
```

Figure 16. stegoVeritas output showing that trailing data is discovered

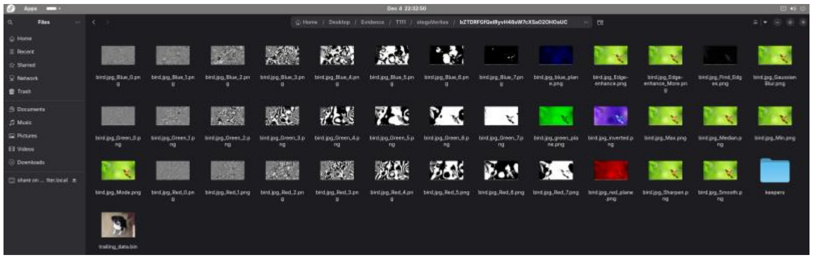


Figure 17. Directory containing the output which includes the trailing data

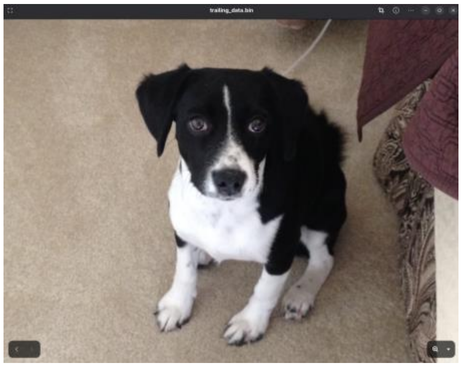
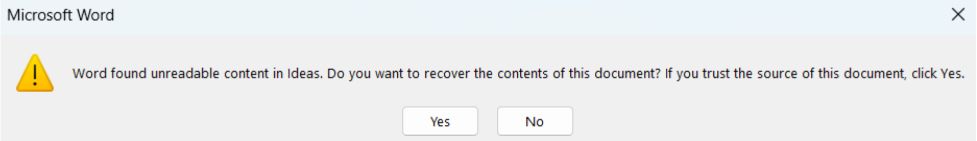


Figure 18. Trailing data of bird.jpg includes an image of the suspect

7. Ideas.docx

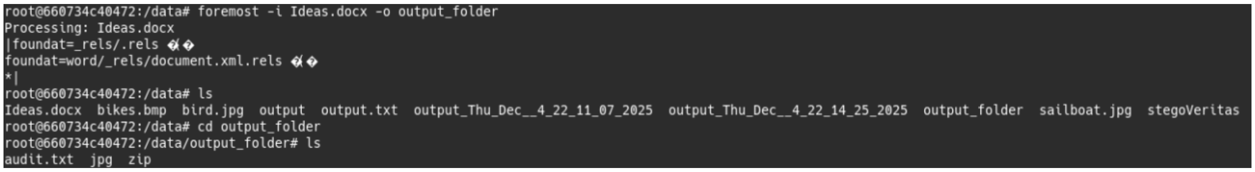
- **Analysis Performed:**
  - The examiner tried to access “Ideas.docx” however Word found unreadable content in the file, as shown in Figure 19. From here, the examiner decided to file carve using foremost.
  - While still in the stego-toolkit container, the examiner used foremost to carve the file of “Ideas.docx”. The command that the examiner used is:
    - *foremost -I Ideas.docx -o output_folder*
  - Within the output folder, “output_folder”, it contains two folders, “jpg” and “zip”, as shown in Figure 20.
- **Answer:**
  The unreadable Microsoft Word file Ideas.docx was determined to contain hidden or corrupted embedded data. Since the document could not be opened normally, the examiner performed file carving using foremost to recover any internal objects. The carving process successfully extracted additional file types from within Ideas.docx. The output directory contained a subfolder, “jpg”, indicating that the document housed an embedded image. Ideas.docx contained a hidden JPEG image of the suspect as shown in Figure 21, successfully recovered through file carving.

- **Supporting Evidence:**



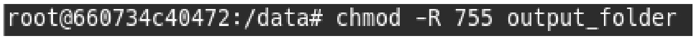
Word found unreadable content in Ideas. Do you want to recover the contents of this document? If you trust the source of this document, click Yes.

Figure 19. Error message after attempt to access "Ideas.docx"



```
root@660734c40472:/data# foremost -i Ideas.docx -o output_folder
Processing: Ideas.docx
|foundat=_rels/.rels ��
foundat=word/_rels/document.xml.rels ��
*|
root@660734c40472:/data# ls
Ideas.docx  bikes.bmp  bird.jpg  output  output.txt  output_Thu_Dec__4_22_11_07_2025  output_Thu_Dec__4_22_14_25_2025  output_folder  sailboat.jpg  stegoVeritas
root@660734c40472:/data# cd output_folder
root@660734c40472:/data/output_folder# ls
audit.txt  jpg  zip
```

Figure 20. The command for foremost



```
root@660734c40472:/data# chmod -R 755 output_folder
```

Figure 21. Changing the permissions of "output_folder" inorder for the examiner to access it using GUI

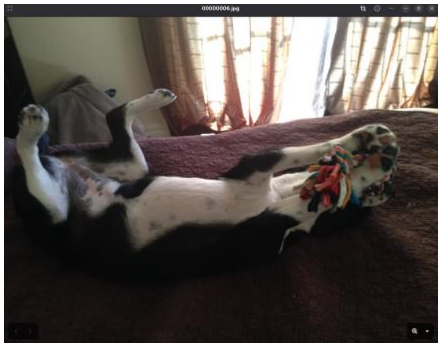
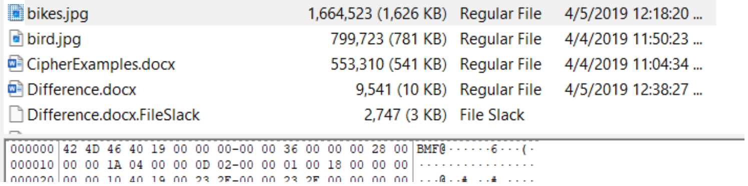


Figure 22. Image of the suspect from within "Ideas.docx"

8. bikes.jpg

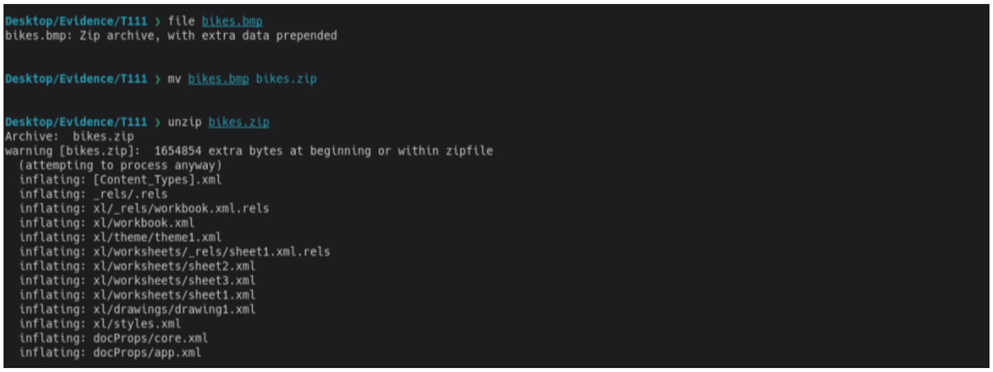
- **Analysis Performed:**
  - Within FTK Imager, the examiner looked at the hex signature of the file and it does not match a JPEG file, as shown in Figure 23. The first bytes of the file are: “42 4D” which corresponds to “BM” in ASCII letters. Therefore, the file is a Windows Bitmap (BMP) file, not a jpeg. The examiner then changed the file extension of “bikes.jpg” to “bikes.bmp”, however upon accessing the file, nothing changed.
  - Then the examiner used the “file” command, as shown in Figure 24. The file command notated that “bikes.bmp” is a ZIP archive. Therefore, the examiner changed the file extension once again to “bikes.zip” and unzipped it, as shown in Figure 24.
  - Then the examiner accessed the unzipped folder and noticed that it resembled as the unzipped folder of an Microsoft Excel file, which it is. The examiner then proceeded to the drawing directory which contained another piece of evidence, as shown in Figure 25.
- **Answer:**
  Forensic review of the file bikes.jpg revealed that it was not an actual JPEG image. Its internal header (“BM”) showed that it was initially a BMP file, but further analysis using the file command confirmed that the file was actually a ZIP archive disguised with multiple misleading extensions. After renaming and extracting the archive, the recovered folder structure matched that of a Microsoft Excel workbook (Office Open XML format). Examination of the internal directories led to the discovery of additional evidence located within the /xl/drawings directory of the reconstructed Excel file. bikes.jpg was a disguised ZIP-based Excel file containing hidden evidence inside its drawings directory, as shown in Figure 25.

- **Supporting Evidence:**



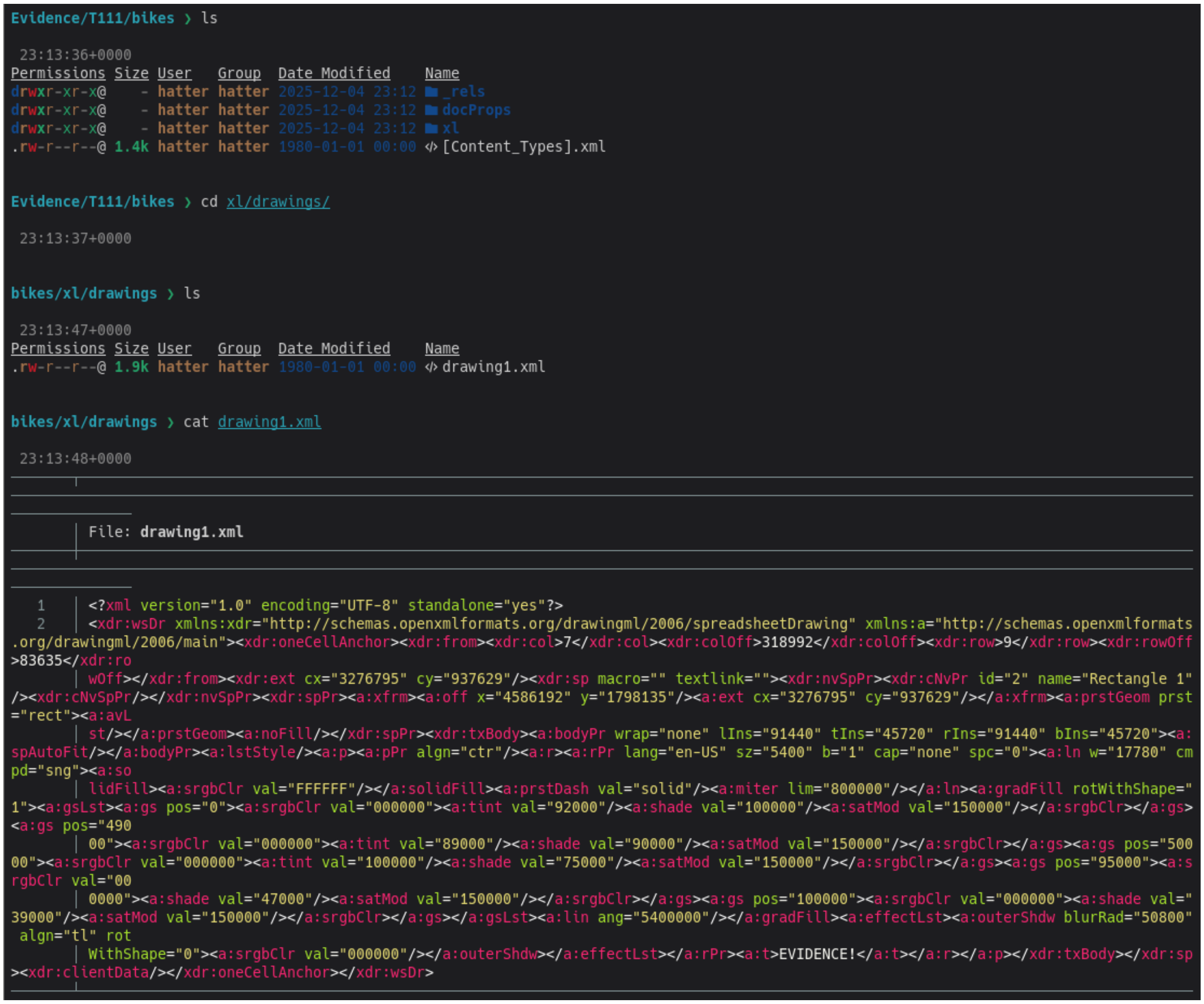
```
bikes.jpg                 1,664,523 (1,626 KB)  Regular File  4/5/2019 12:18:20 ...
bird.jpg                    799,723 (781 KB)    Regular File  4/4/2019 11:50:23 ...
CipherExamples.docx         553,310 (541 KB)    Regular File  4/4/2019 11:04:34 ...
Difference.docx               9,541 (10 KB)     Regular File  4/5/2019 12:38:27 ...
Difference.docx.FileSlack     2,747 (3 KB)      File Slack

000000 42 4D 46 40 19 00 00 00-00 00 36 00 00 00 28 00 BMF@······6···(·
000010 00 00 1A 04 00 00 0D 02-00 00 01 00 18 00 00 00 ················
000020 00 00 10 40 19 00 23 2E-00 00 23 2E 00 00 00 00 ···@··#···#·····
```

Figure 23. Looking at the hex signature of "bikes.jpg" within FTK Imager



```
Desktop/Evidence/T111 › file bikes.bmp
bikes.bmp: Zip archive, with extra data prepended


Desktop/Evidence/T111 › mv bikes.bmp bikes.zip


Desktop/Evidence/T111 › unzip bikes.zip
Archive:  bikes.zip
warning [bikes.zip]:  1654854 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: xl/_rels/workbook.xml.rels
  inflating: xl/workbook.xml
  inflating: xl/theme/theme1.xml
  inflating: xl/worksheets/_rels/sheet1.xml.rels
  inflating: xl/worksheets/sheet2.xml
  inflating: xl/worksheets/sheet3.xml
  inflating: xl/worksheets/sheet1.xml
  inflating: xl/drawings/drawing1.xml
  inflating: xl/styles.xml
  inflating: docProps/core.xml
  inflating: docProps/app.xml
```

Figure 24. Changing the file extension of "bikes.bmp" to "bikes.zip" and then unzipping it



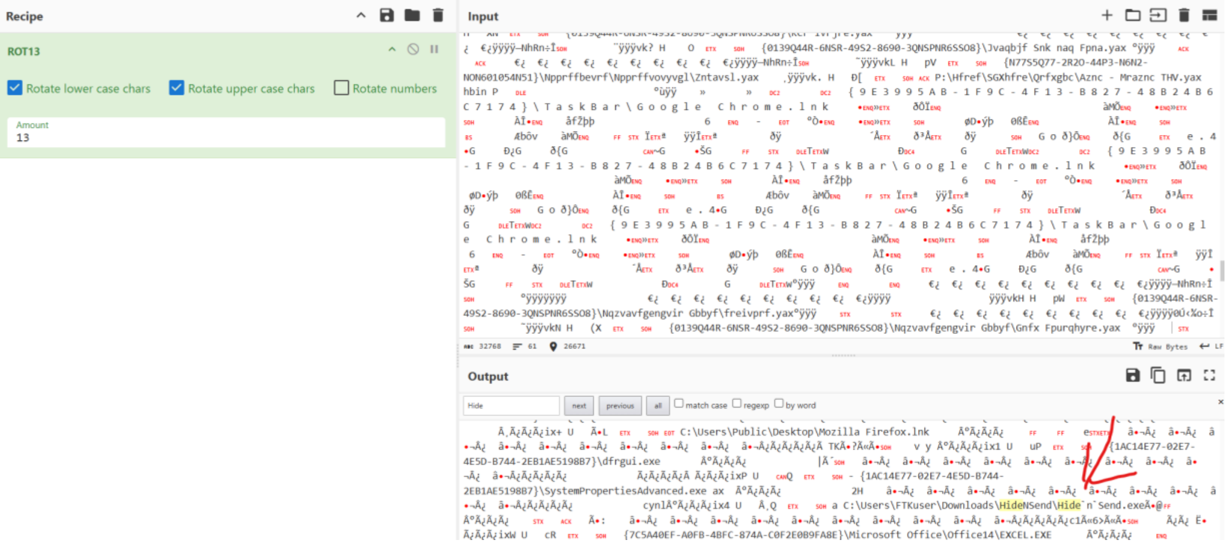
```
Evidence/T111/bikes ❯ ls

 23:13:36+0000
Permissions Size User   Group  Date Modified    Name
drwxr-xr-x@    - hatter hatter 2025-12-04 23:12 📁 _rels
drwxr-xr-x@    - hatter hatter 2025-12-04 23:12 📁 docProps
drwxr-xr-x@    - hatter hatter 2025-12-04 23:12 📁 xl
.rw-r--r--@ 1.4k hatter hatter 1980-01-01 00:00 </> [Content_Types].xml


Evidence/T111/bikes ❯ cd xl/drawings/

 23:13:37+0000


bikes/xl/drawings ❯ ls

 23:13:47+0000
Permissions Size User   Group  Date Modified    Name
.rw-r--r--@ 1.9k hatter hatter 1980-01-01 00:00 </> drawing1.xml


bikes/xl/drawings ❯ cat drawing1.xml

 23:13:48+0000

        │ File: drawing1.xml

        │ <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        │ <xdr:wsDr xmlns:xdr="http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing" xmlns:a="http://schemas.openxmlformats
.org/drawingml/2006/main"><xdr:oneCellAnchor><xdr:from><xdr:col>7</xdr:col><xdr:colOff>318992</xdr:colOff><xdr:row>9</xdr:row><xdr:rowOff
>83635</xdr:ro
        │ wOff></xdr:from><xdr:ext cx="3276795" cy="937629"/><xdr:sp macro="" textlink=""><xdr:nvSpPr><xdr:cNvPr id="2" name="Rectangle 1"
/><xdr:cNvSpPr/></xdr:nvSpPr><xdr:spPr><a:xfrm><a:off x="4586192" y="1798135"/><a:ext cx="3276795" cy="937629"/></a:xfrm><a:prstGeom prst
="rect"><a:avL
        │ st/></a:prstGeom><a:noFill/></xdr:spPr><xdr:txBody><a:bodyPr wrap="none" lIns="91440" tIns="45720" rIns="91440" bIns="45720"><a:
spAutoFit/></a:bodyPr><a:lstStyle/><a:p><a:pPr algn="ctr"/><a:r><a:rPr lang="en-US" sz="5400" b="1" cap="none" spc="0"><a:ln w="17780" cm
pd="sng"><a:so
        │ lidFill><a:srgbClr val="FFFFFF"/></a:solidFill><a:prstDash val="solid"/><a:miter lim="800000"/></a:ln><a:gradFill rotWithShape="
1"><a:gsLst><a:gs pos="0"><a:srgbClr val="000000"><a:tint val="92000"/><a:shade val="100000"/><a:satMod val="150000"/></a:srgbClr></a:gs>
<a:gs pos="490
        │ 00"><a:srgbClr val="000000"><a:tint val="89000"/><a:shade val="90000"/><a:satMod val="150000"/></a:srgbClr></a:gs><a:gs pos="500
00"><a:srgbClr val="000000"><a:tint val="100000"/><a:shade val="75000"/><a:satMod val="150000"/></a:srgbClr></a:gs><a:gs pos="95000"><a:s
rgbClr val="00
        │ 0000"><a:shade val="47000"/><a:satMod val="150000"/></a:srgbClr></a:gs><a:gs pos="100000"><a:srgbClr val="000000"><a:shade val="
39000"/><a:satMod val="150000"/></a:srgbClr></a:gs></a:gsLst><a:lin ang="5400000"/></a:gradFill><a:effectLst><a:outerShdw blurRad="50800"
 algn="tl" rot
        │ WithShape="0"><a:srgbClr val="000000"/></a:outerShdw></a:effectLst></a:rPr><a:t>EVIDENCE!</a:t></a:r></a:p></xdr:txBody></xdr:sp
><xdr:clientData/></xdr:oneCellAnchor></xdr:wsDr>
```

Figure 25. Finding EVIDENCE within the unzipped contents of "bikes.zip"

9. sailboat.jpg

- **Analysis Performed:**
  - The examiner opened the UserAssist file and notated that it may be ROT13 encrypted, the examiner then proceeded to CyberChef and applied a ROT13 decryption mode, as shown in Figure 26. The examiner went through all the applications that the user used and there was an application that stood out called, “Hide’N’Seek”, as shown in Figure 26.
  - The examiner then went to download Hide’N’Seek from the internet, as shown in Figure 27.
  - The examiner then inputted the “sailboat.jpg” file into Hide’N’Seek with the extraction directory set to an evidence folder, as shown in Figure 28. There was a password needed to extract the file and the examiner kept guessing commonly used passwords and ended up with the correct password of “123456”, as shown in Figure 28.
  - The examiner then accessed the extracted file, “SteggedEvidence.txt”, which contained another piece of EVIDENCE, as shown in Figure 29
- **Answer:**
  Analysis of the UserAssist artifacts (after applying ROT13 decoding) revealed that the user had executed an application named “Hide’N’Seek.” This strongly suggested that the suspect used that program to conceal data. The examiner obtained the same tool, loaded sailboat.jpg into Hide’N’Seek, and successfully extracted hidden content after guessing the password “123456.” The extraction produced a text file named “SteggedEvidence.txt” in the designated evidence folder, which contained an additional hidden message identified as EVIDENCE in this case, as shown in Figure 29.

- **Supporting Evidence:**



ROT13

☑ Rotate lower case chars ☑ Rotate upper case chars □ Rotate numbers

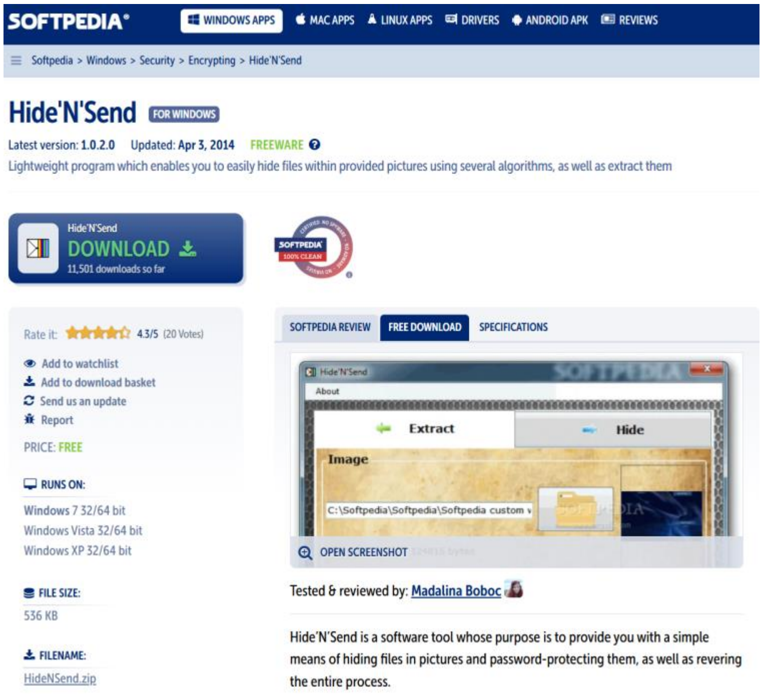
Amount
13

```
{0139Q44R-6NSR-49S2-8690-3QNSPNR6SSO8}\Nqzvavfgengvir Gbbyf\Gnfx Fpurqhyre.yax
32768  61  26671
```

Hide □ match case □ regexp □ by word

```
C:\Users\Public\Desktop\Mozilla Firefox.lnk
{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\dfrgui.exe
{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\SystemPropertiesAdvanced.exe
C:\Users\FTKuser\Downloads\HideNSend\Hide`n`Send.exe
{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Microsoft Office\Office14\EXCEL.EXE
```

Figure 26. Decoded UserAssist output showing "HideNSend.exe" being used by the user



Softpedia > Windows > Security > Encrypting > Hide'N'Send

Hide'N'Send FOR WINDOWS

Latest version: 1.0.2.0 Updated: Apr 3, 2014 FREEWARE

Lightweight program which enables you to easily hide files within provided pictures using several algorithms, as well as extract them

Hide'N'Send DOWNLOAD 11,501 downloads so far

Rate it: 4.3/5 (20 Votes)

PRICE: FREE

RUNS ON:

Windows 7 32/64 bit
Windows Vista 32/64 bit
Windows XP 32/64 bit

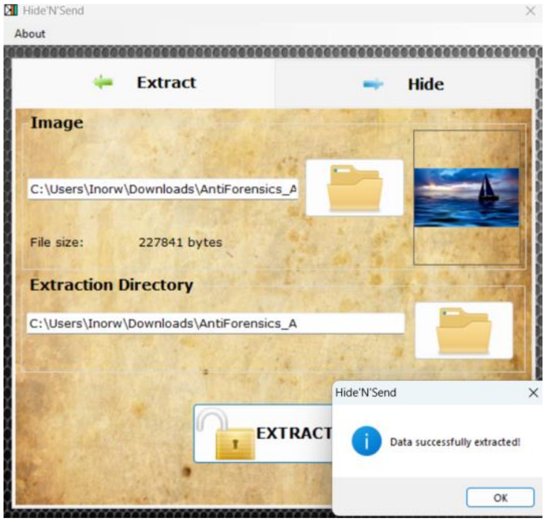
FILE SIZE:
536 KB

FILENAME:
HideNSend.zip

Tested & reviewed by: Madalina Boboc

Hide'N'Send is a software tool whose purpose is to provide you with a simple means of hiding files in pictures and password-protecting them, as well as revering the entire process.

Figure 27. Downloading Hide'N'Send on Google



Image

C:\Users\Inorw\Downloads\AntiForensics_A

File size: 227841 bytes

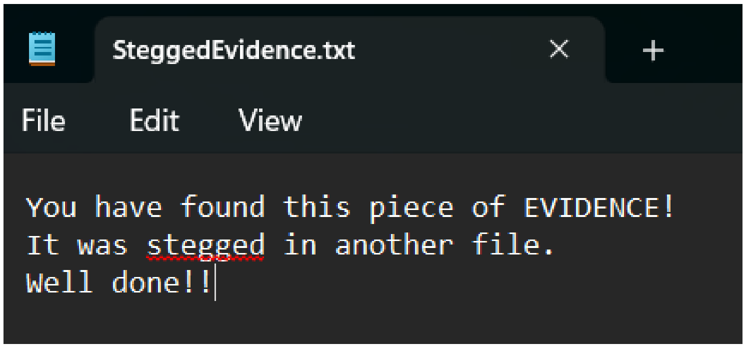
Extraction Directory

C:\Users\Inorw\Downloads\AntiForensics_A

Hide'N'Send

Data successfully extracted!

Figure 28. Inputting the "sailboat.jpg" into "Hide'N'Seek" and successfully extracted



```
You have found this piece of EVIDENCE!
It was stegged in another file.
Well done!!
```

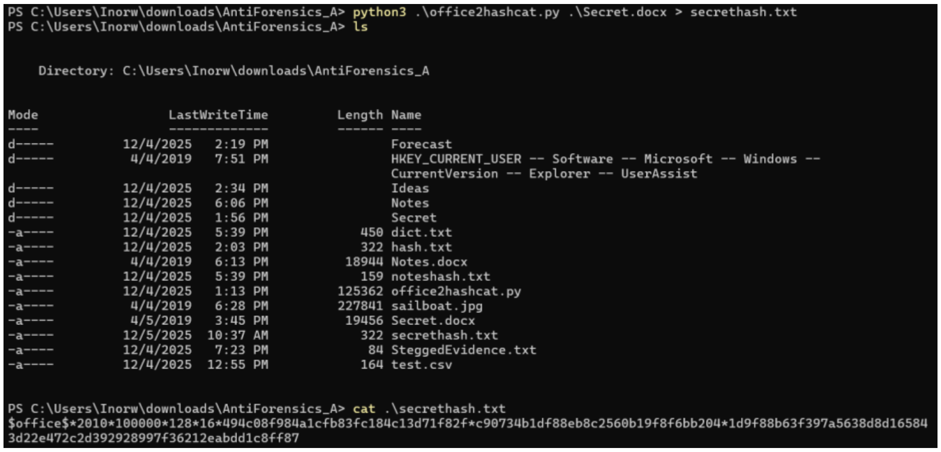
Figure 29. Accessing the outputted extracted file that was in "sailboat.jpg"

10. Secret.docx

- **Analysis Performed:**
  - While in the AntiForensics_A directory, the examiner ran python3 .\office2hashcat.py Secret.docx > hash.txt to extract the password hash from the protected Secret.docx file into secrethash.txt, as shown in Figure 30.
  - The examiner viewed hash.txt and confirmed that the hash string started with $office$*2010*…, indicating a Microsoft Office 2010 document, as shown in Figure 30.
  - The examiner then consulted Hashcat's hash-mode list and confirmed that mode 9500 corresponds to "MS Office 2010", which is the correct mode for this hash, as shown in Figure 31.
  - A custom dictionary file (dict.txt) was prepared from biographical terms and other words identified in earlier evidence, as shown in Figure 32.
  - From the Hashcat installation directory, the examiner launched a combinator attack using mode 9500, the extracted Office hash, and the custom dictionary on both sides, with a rule to join two dictionary words (e.g., adding a separator between them).
    - Example command (simplified): *hashcat.exe -a 1 -m 9500 -o found.txt secrethash.txt dict.txt -j '$-' dict.txt*
  - Hashcat successfully cracked the Office hash and wrote the result to found.txt, as shown in Figure 33.
  - Inspecting found.txt showed that the recovered password was a combination of the words "mazda" and "amherst", as shown in Figure 34.
  - Using this recovered password, the examiner opened Secret.docx, which contained the final piece of EVIDENCE stating that the user had to decipher two different files to obtain the information for the biographical dictionary, as shown in Figure 35.
- **Answer:**

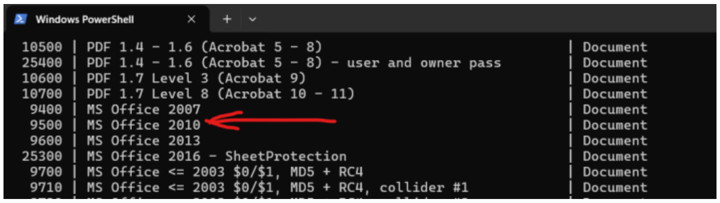
  By extracting the password hash from the protected Secret.docx file and identifying it as an MS Office 2010 hash (mode 9500 in Hashcat), the examiner was able to target the correct hash type and run a combinator attack using a custom dictionary built from biographical terms discovered elsewhere in the case. This attack successfully recovered the password "mazda_amherst". When the examiner used this password to open Secret.docx, the document revealed the final piece of EVIDENCE, as shown in Figure 34.

- **Supporting Evidence:**



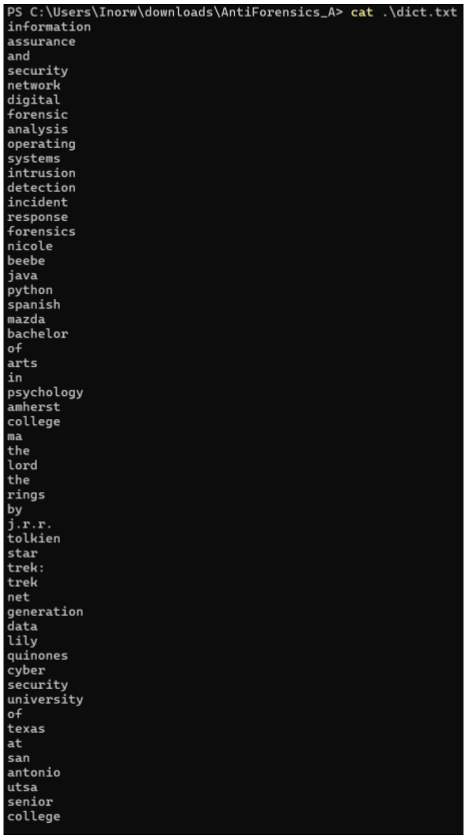
```
PS C:\Users\Inorw\downloads\AntiForensics_A> python3 .\office2hashcat.py .\Secret.docx > secrethash.txt
PS C:\Users\Inorw\downloads\AntiForensics_A> ls


    Directory: C:\Users\Inorw\downloads\AntiForensics_A


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         12/4/2025   2:19 PM                Forecast
d-----          4/4/2019   7:51 PM                HKEY_CURRENT_USER -- Software -- Microsoft -- Windows --
                                                  CurrentVersion -- Explorer -- UserAssist
d-----         12/4/2025   2:34 PM                Ideas
d-----         12/4/2025   6:06 PM                Notes
d-----         12/4/2025   1:56 PM                Secret
-a----         12/4/2025   5:39 PM            450 dict.txt
-a----         12/4/2025   2:03 PM            322 hash.txt
-a----          4/4/2019   6:13 PM          18944 Notes.docx
-a----         12/4/2025   5:39 PM            159 noteshash.txt
-a----         12/4/2025   1:13 PM         125362 office2hashcat.py
-a----          4/4/2019   6:28 PM         227841 sailboat.jpg
-a----          4/5/2019   3:45 PM          19456 Secret.docx
-a----         12/5/2025  10:37 AM            322 secrethash.txt
-a----         12/4/2025   7:23 PM             84 SteggedEvidence.txt
-a----         12/4/2025  12:55 PM            164 test.csv


PS C:\Users\Inorw\downloads\AntiForensics_A> cat .\secrethash.txt
$office$*2010*100000*128*16*494c08f984a1cfb83fc184c13d71f82f*c90734b1df88eb8c2560b19f8f6bb204*1d9f88b63f397a5638d8d16584
3d22e472c2d392928997f36212eabdd1c8ff87
```

Figure 30. PowerShell output showing the examiner using office2hashcat.py to convert Secret.docx into a Hashcat-compatible Office hash, listing the contents of the AntiForensics_A directory, and confirming the extracted $office$ hash saved in secrethash.txt.



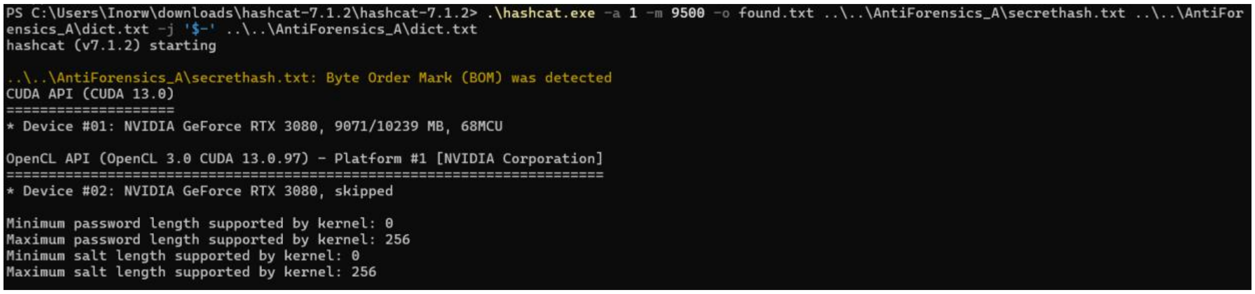
```
  10500 | PDF 1.4 - 1.6 (Acrobat 5 - 8)                              | Document
  25400 | PDF 1.4 - 1.6 (Acrobat 5 - 8) - user and owner pass        | Document
  10600 | PDF 1.7 Level 3 (Acrobat 9)                                | Document
  10700 | PDF 1.7 Level 8 (Acrobat 10 - 11)                          | 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
```

Figure 31. Hashcat hash-mode reference output showing that mode 9500 corresponds to "MS Office 2010 – Document", confirming the correct mode for the extracted Secret.docx hash.



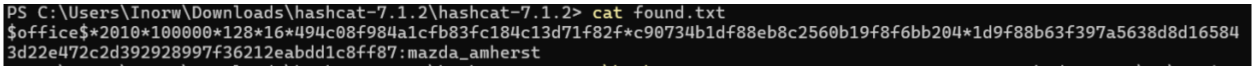
```
PS C:\Users\Inorw\downloads\AntiForensics_A> cat .\dict.txt
information
assurance
and
security
network
digital
forensic
analysis
operating
systems
intrusion
detection
incident
response
forensics
nicole
beebe
java
python
spanish
mazda
bachelor
of
arts
in
psychology
amherst
college
ma
the
lord
the
rings
by
j.r.r.
tolkien
star
trek:
trek
net
generation
data
lily
quinones
cyber
security
university
of
texas
at
san
antonio
utsa
senior
college
```

Figure 32. Biographical dictionary based off of knowledge on Lily



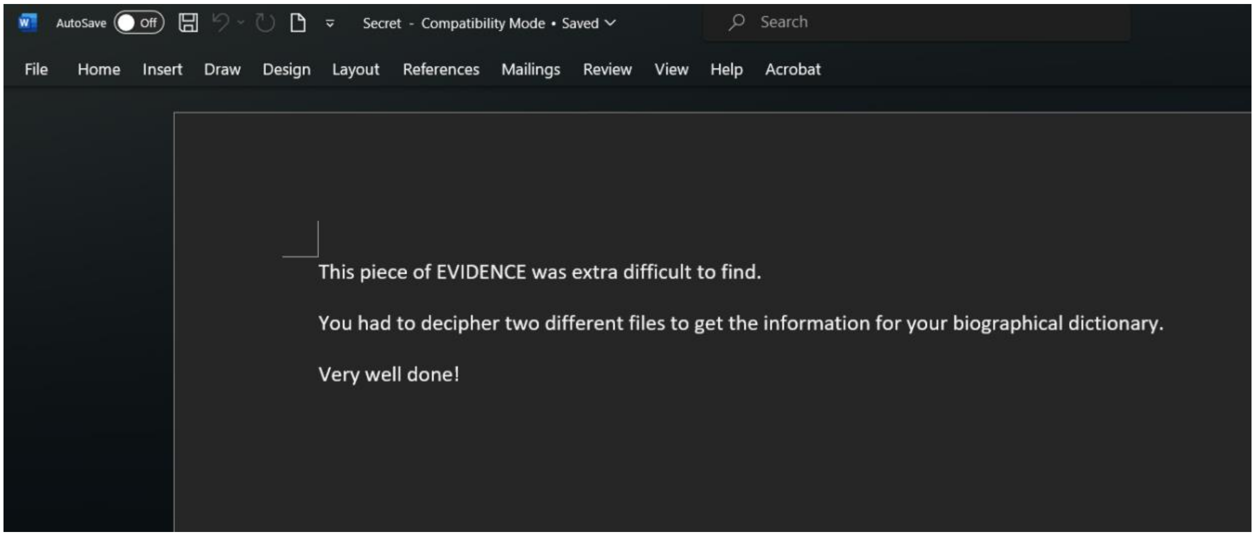
```
PS C:\Users\Inorw\downloads\hashcat-7.1.2\hashcat-7.1.2> .\hashcat.exe -a 1 -m 9500 -o found.txt ..\..\AntiForensics_A\secrethash.txt ..\..\AntiFor
ensics_A\dict.txt -j '$-' ..\..\AntiForensics_A\dict.txt
hashcat (v7.1.2) starting

..\..\AntiForensics_A\secrethash.txt: Byte Order Mark (BOM) was detected
CUDA API (CUDA 13.0)
====================
* Device #01: NVIDIA GeForce RTX 3080, 9071/10239 MB, 68MCU

OpenCL API (OpenCL 3.0 CUDA 13.0.97) - Platform #1 [NVIDIA Corporation]
=======================================================================
* Device #02: NVIDIA GeForce RTX 3080, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256
```

Figure 33. Hashcat session using attack mode -a 1 and hash-mode 9500 with the custom dict.txt wordlist (and join rule -j '$-') to brute-force the Secrert.docx password.



```
PS C:\Users\Inorw\Downloads\hashcat-7.1.2\hashcat-7.1.2> cat found.txt
$office$*2010*100000*128*16*494c08f984a1cfb83fc184c13d71f82f*c90734b1df88eb8c2560b19f8f6bb204*1d9f88b63f397a5638d8d16584
3d22e472c2d392928997f36212eabdd1c8ff87:mazda_amherst
```

Figure 34. Contents of found.txt after the Hashcat run, showing the cracked MS Office 2010 hash from Secret.docx and the recovered password "mazda_amherst."



This piece of EVIDENCE was extra difficult to find.

You had to decipher two different files to get the information for your biographical dictionary.

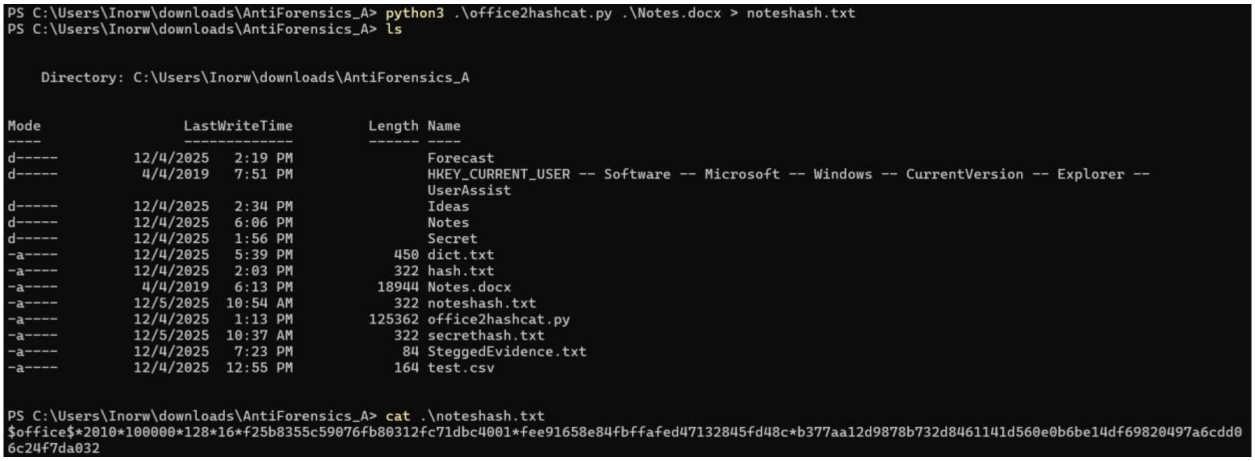
Very well done!

Figure 35. Decrypted Secret.docx opened in Microsoft Word, displaying the final EVIDENCE message confirming that two different files had to be deciphered to obtain the biographical dictionary information.

11. Notes.docx

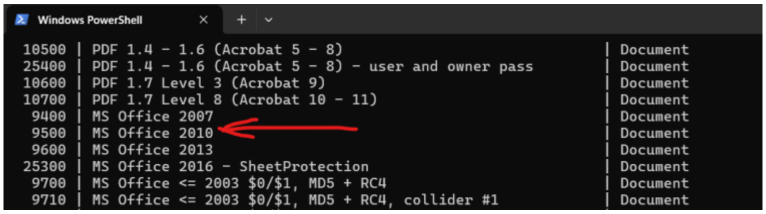
- **Analysis Performed:**
  - While in the AntiForensics_A directory, the examiner ran python3 .\office2hashcat.py .\Notes.docx > noteshash.txt to extract the password hash from the protected Notes.docx file into noteshash.txt, as shown in Figure 36.
  - The examiner viewed noteshash.txt and confirmed that the hash string began with $office$*2010*…, indicating that Notes.docx is a Microsoft Office 2010 document, as shown in Figure 36.
  - The examiner consulted Hashcat's hash-mode list and verified that mode 9500 corresponds to "MS Office 2010," confirming the correct mode for this hash, as shown in Figure 37.
  - The examiner inspected dict.txt, a custom dictionary file built from biographical and case-related terms identified in earlier analysis, as shown in Figure 38.
  - From the Hashcat installation directory, the examiner launched a combinator attack using mode 9500 against noteshash.txt, using dict.txt as the wordlist for both sides of the combination.
    - Example command (simplified): hashcat.exe -a 1 -m 9500 -o found.txt ..\..\AntiForensics_A\noteshash.txt ..\..\AntiForensics_A\dict.txt (Figure 39).
  - Hashcat successfully cracked the Office hash and wrote the recovered credential pair to found.txt, as shown in Figure 39.
  - Reviewing found.txt showed that the recovered password for Notes.docx was quinonesutsa, as shown in Figure 40.
  - Using the recovered password, the examiner opened Notes.docx, which contained a message confirming that the password had been successfully cracked and identifying the document as EVIDENCE, as shown in Figure 41.
- **Answer:**
  While in the AntiForensics_A directory, the examiner used python3 .\office2hashcat.py .\Notes.docx > noteshash.txt to extract the password hash from the protected Notes.docx file, confirming that the resulting hash in noteshash.txt began with $office$*2010*…, consistent with a Microsoft Office 2010 document (Figure 36). The examiner then consulted Hashcat's hash-mode reference and verified that mode 9500 is designated for "MS Office 2010," confirming the correct cracking mode for this hash (Figure 37). A custom wordlist, dict.txt, compiled from biographical and case-related terms identified earlier in the investigation, was reviewed and selected as the dictionary for the attack (Figure 38). From the Hashcat installation directory, the examiner ran a combinator attack using mode 9500 against noteshash.txt with dict.txt as the wordlist, directing successful cracks to found.txt (Figure 39). Hashcat successfully recovered a credential, and inspection of found.txt showed that the password for Notes.docx was "quinonesutsa" (Figure 40). Using this password, the examiner opened Notes.docx, which displayed a confirmation message stating that the password had been cracked and explicitly identifying the document as EVIDENCE (Figure 41).

- **Supporting Evidence:**



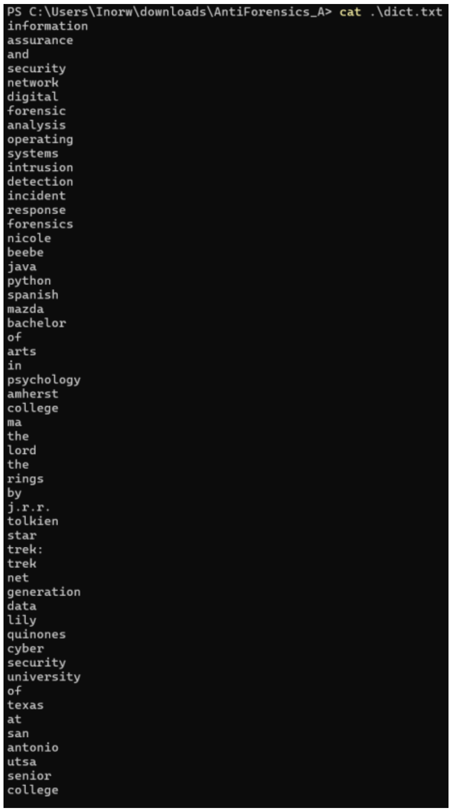
```
PS C:\Users\Inorw\downloads\AntiForensics_A> python3 .\office2hashcat.py .\Notes.docx > noteshash.txt
PS C:\Users\Inorw\downloads\AntiForensics_A> ls


    Directory: C:\Users\Inorw\downloads\AntiForensics_A


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         12/4/2025   2:19 PM                Forecast
d-----          4/4/2019   7:51 PM                HKEY_CURRENT_USER -- Software -- Microsoft -- Windows -- CurrentVersion -- Explorer --
                                                  UserAssist
d-----         12/4/2025   2:34 PM                Ideas
d-----         12/4/2025   6:06 PM                Notes
d-----         12/4/2025   1:56 PM                Secret
-a----         12/4/2025   5:39 PM            450 dict.txt
-a----         12/4/2025   2:03 PM            322 hash.txt
-a----          4/4/2019   6:13 PM          18944 Notes.docx
-a----         12/5/2025  10:54 AM            322 noteshash.txt
-a----         12/4/2025   1:13 PM         125362 office2hashcat.py
-a----         12/5/2025  10:37 AM            322 secrethash.txt
-a----         12/4/2025   7:23 PM             84 SteggedEvidence.txt
-a----         12/4/2025  12:55 PM            164 test.csv


PS C:\Users\Inorw\downloads\AntiForensics_A> cat .\noteshash.txt
$office$*2010*100000*128*16*f25b8355c59076fb80312fc71dbc4001*fee91658e84fbffafed47132845fd48c*b377aa12d9878b732d8461141d560e0b6be14df69820497a6cdd0
6c24f7da032
```

Figure 36. PowerShell output showing office2hashcat.py generating noteshash.txt from Notes.docx and displaying the resulting MS Office hash.



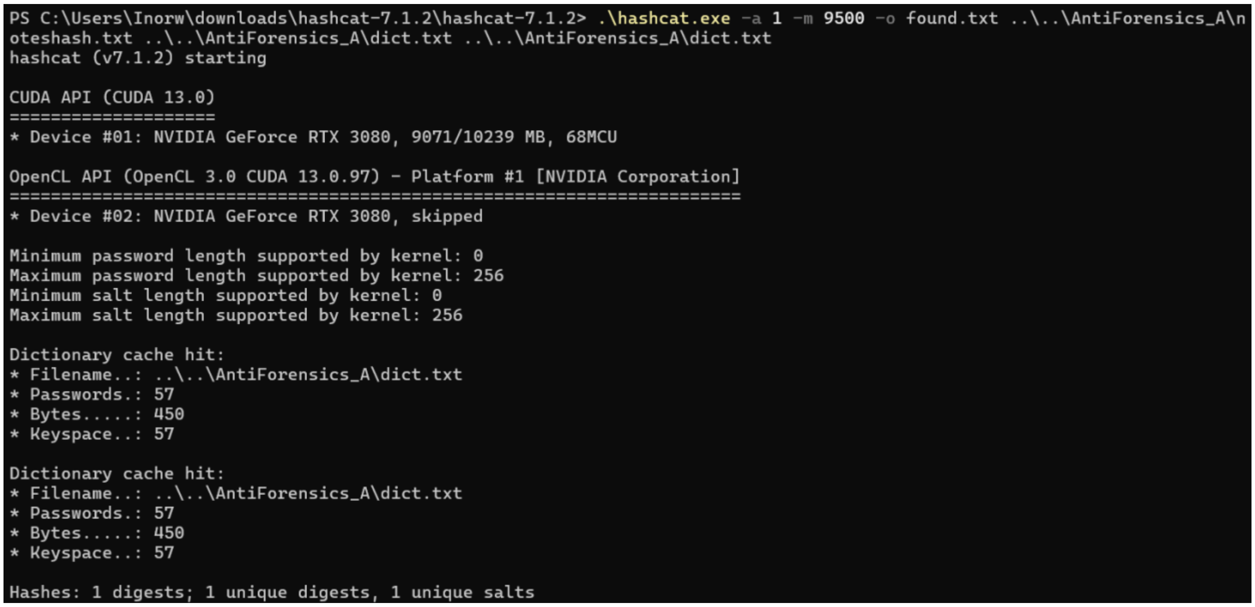
```
  10500 | PDF 1.4 - 1.6 (Acrobat 5 - 8)                              | Document
  25400 | PDF 1.4 - 1.6 (Acrobat 5 - 8) - user and owner pass        | Document
  10600 | PDF 1.7 Level 3 (Acrobat 9)                                | Document
  10700 | PDF 1.7 Level 8 (Acrobat 10 - 11)                          | 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
```

Figure 37. Excerpt from Hashcat's hash-mode list highlighting mode 9500 – MS Office 2010 used for the attack.



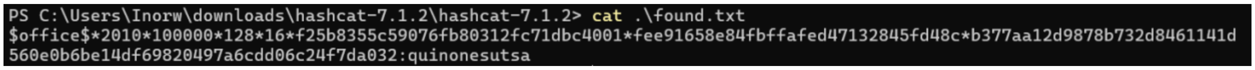
```
PS C:\Users\Inorw\downloads\AntiForensics_A> cat .\dict.txt
information
assurance
and
security
network
digital
forensic
analysis
operating
systems
intrusion
detection
incident
response
forensics
nicole
beebe
java
python
spanish
mazda
bachelor
of
arts
in
psychology
amherst
college
ma
the
lord
the
rings
by
j.r.r.
tolkien
star
trek:
trek
net
generation
data
lily
quinones
cyber
security
university
of
texas
at
san
antonio
utsa
senior
college
```

Figure 38. Contents of dict.txt, the custom wordlist used as the dictionary for the password-cracking attempt.



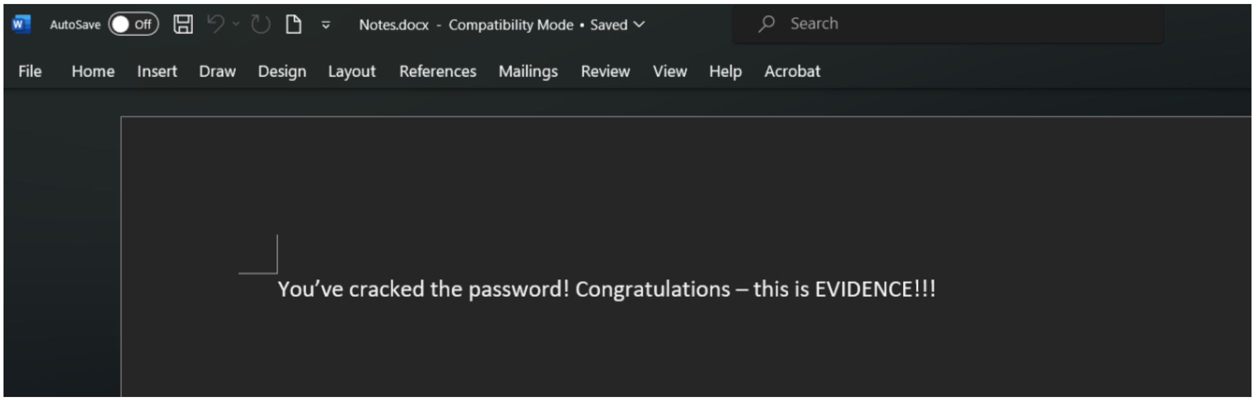
```
PS C:\Users\Inorw\downloads\hashcat-7.1.2\hashcat-7.1.2> .\hashcat.exe -a 1 -m 9500 -o found.txt ..\..\AntiForensics_A\n
oteshash.txt ..\..\AntiForensics_A\dict.txt ..\..\AntiForensics_A\dict.txt
hashcat (v7.1.2) starting

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

OpenCL API (OpenCL 3.0 CUDA 13.0.97) - Platform #1 [NVIDIA Corporation]
========================================================================
* Device #02: NVIDIA GeForce RTX 3080, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Dictionary cache hit:
* Filename..: ..\..\AntiForensics_A\dict.txt
* Passwords.: 57
* Bytes.....: 450
* Keyspace..: 57

Dictionary cache hit:
* Filename..: ..\..\AntiForensics_A\dict.txt
* Passwords.: 57
* Bytes.....: 450
* Keyspace..: 57

Hashes: 1 digests; 1 unique digests, 1 unique salts
```

Figure 39. Hashcat run using mode 9500 and dict.txt against noteshash.txt, indicating one hash loaded and processed successfully.



```
PS C:\Users\Inorw\downloads\hashcat-7.1.2\hashcat-7.1.2> cat .\found.txt
$office$*2010*100000*128*16*f25b8355c59076fb80312fc71dbc4001*fee91658e84fbffafed47132845fd48c*b377aa12d9878b732d8461141d
560e0b6be14df69820497a6cdd06c24f7da032:quinonesutsa
```

Figure 40. PowerShell view of found.txt, showing the recovered password quinonesutsa appended to the MS Office 2010 hash entry.



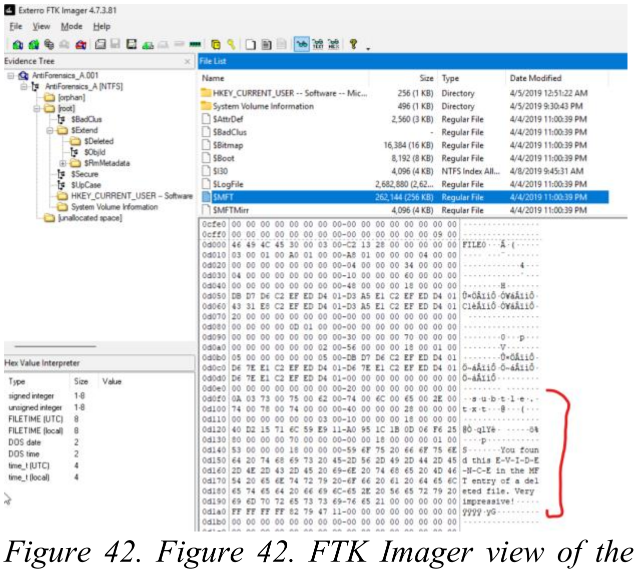
You've cracked the password! Congratulations – this is EVIDENCE!!!

Figure 41. Decrypted Notes.docx opened in Microsoft Word, displaying the message "You've cracked the password! Congratulations – this is EVIDENCE!!!"

12. $MFT

- **Analysis Performed:**
    - The examiner launched FTK Imager and added the extracted AntiForensics_A folder as an evidence item so that its NTFS metadata files could be reviewed.
    - Within the Evidence Tree, the examiner navigated to the NTFS root directory and selected the special metadata file $MFT (Master File Table) for detailed examination.
    - Using FTK Imager's hex/text viewer, the examiner scrolled through the $MFT contents and examined the ASCII pane on the right side for human-readable strings.
    - In one $MFT entry associated with a deleted file, the examiner observed the embedded message: "You found this E-V-I-D-E-N-C-E in the MFT entry of a deleted file. Very impressive!", confirming that the required EVIDENCE was intentionally hidden within the $MFT of the volume.
- **Answer:**
  In FTK Imager, the examiner first added the recovered AntiForensics_A folder as an evidence item, enabling direct access to the volume's NTFS metadata files. From the Evidence Tree, the examiner navigated to the root of the NTFS file system and selected the special $MFT file, which serves as the Master File Table containing an entry for every file and directory on the volume, including deleted ones. Each $MFT entry can store file attributes, timestamps, and residual data, making it a valuable source of hidden or anti-forensic content. Using FTK Imager's hex and ASCII viewer, the examiner scrolled through the $MFT and inspected the ASCII column for readable strings. Within one entry corresponding to a deleted file, the examiner identified a clear message stating that the examiner had found "E-V-I-D-E-N-C-E in the MFT entry of a deleted file," thereby confirming that this $MFT record contained the intended evidence, as shown in Figure 42.

- **Supporting Evidence:**



Name	Size	Type	Date Modified
HKEY_CURRENT_USER -- Software -- Mic...	256 (1 KB)	Directory	4/5/2019 12:51:22 AM
System Volume Information	496 (1 KB)	Directory	4/5/2019 9:30:43 PM
$AttrDef	2,560 (3 KB)	Regular File	4/4/2019 11:00:39 PM
$BadClus	-	Regular File	4/4/2019 11:00:39 PM
$Bitmap	16,384 (16 KB)	Regular File	4/4/2019 11:00:39 PM
$Boot	8,192 (8 KB)	Regular File	4/4/2019 11:00:39 PM
$I30	4,096 (4 KB)	NTFS Index All...	4/8/2019 9:45:31 AM
$LogFile	2,682,880 (2,62...	Regular File	4/4/2019 11:00:39 PM
$MFT	262,144 (256 KB)	Regular File	4/4/2019 11:00:39 PM
$MFTMirr	4,096 (4 KB)	Regular File	4/4/2019 11:00:39 PM

```
0cfe0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
0cff0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 09 00  ················
0d000  46 49 4C 45 30 00 03 00-C2 13 28 00 00 00 00 00  FILE0···Â·(·····
0d010  03 00 01 00 A0 01 00 00-A8 01 00 00 00 04 00 00  ········¨·······
0d020  00 00 00 00 00 00 00 00-04 00 00 00 34 00 00 00  ············4···
0d030  04 00 00 00 00 00 00 00-10 00 00 00 60 00 00 00  ············`···
0d040  00 00 00 00 00 00 00 00-48 00 00 00 18 00 00 00  ········H·······
0d050  DB D7 D6 C2 EF ED D4 01-D3 A5 E1 C2 EF ED D4 01  Û×ÖÂïíÔ·Ó¥áÂïíÔ·
0d060  43 31 E8 C2 EF ED D4 01-D3 A5 E1 C2 EF ED D4 01  C1èÂïíÔ·Ó¥áÂïíÔ·
0d070  20 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ···············
0d080  00 00 00 00 0D 01 00 00-00 00 00 00 00 00 00 00  ················
0d090  00 00 00 00 00 00 00 00-30 00 00 00 70 00 00 00  ········0···p···
0d0a0  00 00 00 00 00 00 02 00-56 00 00 00 18 00 01 00  ········V·······
0d0b0  05 00 00 00 00 00 05 00-DB D7 D6 C2 EF ED D4 01  ········Û×ÖÂïíÔ·
0d0c0  D6 7E E1 C2 EF ED D4 01-D6 7E E1 C2 EF ED D4 01  Ö~áÂïíÔ·Ö~áÂïíÔ·
0d0d0  D6 7E E1 C2 EF ED D4 01-00 00 00 00 00 00 00 00  Ö~áÂïíÔ·········
0d0e0  00 00 00 00 00 00 00 00-20 00 00 00 00 00 00 00  ········ ·······
0d0f0  0A 03 73 00 75 00 62 00-74 00 6C 00 65 00 2E 00  ··s·u·b·t·l·e·.·
0d100  74 00 78 00 74 00 00 00-40 00 00 00 28 00 00 00  t·x·t···@···(···
0d110  00 00 00 00 00 00 03 00-10 00 00 00 18 00 00 00  ················
0d120  40 D2 15 71 6C 59 E9 11-A0 95 1C 1B 0D 06 F6 25  @Ò·qlYé·········ö%
0d130  80 00 00 00 70 00 00 00-00 00 18 00 00 00 01 00  ····p···········
0d140  53 00 00 00 18 00 00 00-59 6F 75 20 66 6F 75 6E  S·······You foun
0d150  64 20 74 68 69 73 20 45-2D 56 2D 49 2D 44 2D 45  d this E-V-I-D-E
0d160  2D 4E 2D 43 2D 45 20 69-6E 20 74 68 65 20 4D 46  -N-C-E in the MF
0d170  54 20 65 6E 74 72 79 20-6F 66 20 61 20 64 65 6C  T entry of a del
0d180  65 74 65 64 20 66 69 6C-65 2E 20 56 65 72 79 20  eted file. Very 
0d190  69 6D 70 72 65 73 73 69-76 65 21 00 00 00 00 00  impressive!·····
0d1a0  FF FF FF FF 82 79 47 11-00 00 00 00 00 00 00 00  ÿÿÿÿ·yG·········
0d1b0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

Type	Size	Value
signed integer	1-8	
unsigned integer	1-8	
FILETIME (UTC)	8	
FILETIME (local)	8	
DOS date	2	
DOS time	2	
time_t (UTC)	4	
time_t (local)	4	

Figure 42. Figure 42. FTK Imager view of the NTFS $MFT file, showing an ASCII message revealing the word "EVIDENCE" embedded in an entry for a deleted file.

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.

Reddy, N. (2019). Practical cyber forensics: An incident-based approach to forensic investigations. APress. https://doi.org/10.1007/978-1-4842-4460-9.

VeraCrypt Project. (2025). *VeraCrypt*. https://veracrypt.io/en/Downloads.html.