eyehatemalwares

```
root@root:->  vi Blog.txt
-
-       Blog: https://eyehatemalwares.com
-:wq

root@root:->  vi Mission.txt
-
-       To Provide free quality education regarding
-       Blue Teaming and guide startup individuals
-       to identify their passion and improved their
-       skill set in the field of Cybersecurity.
-:wq

root@root:->  vi Vision.txt
-
-       With quality content and education, We
-       will make the future of cyberspace a safe
-       environment for every individual.
-:wq

root@root:-> vi DISCLAIMER.txt
-
-       The opinions expressed in this blog are my own views,
-       and I have based my review on my personal research. All the
-       content in this blog is completely original; this might mention
-       any other blog post, magazine article, any book, and there-
-       fore I regret any resemblance with any of the aforementioned,
-       and would like to assure you that it was unintentional. Moreover,
-       I do not intend to be offensive towards anyone who read this blog,
-       if anything written can be perceived as hurtful to any community
-       or person, I apologize, but that was not the sole purpose of writing
-       it.
-:wq
```

Table of Contents

Phishing Incident Detection and Response:
Identifying Email and Document Existence
using Memory Forensics

I.I - Scenario

Due to fear of getting punished, an employee trashed a possible Phishing email after she/he clicked and downloaded the attachment.

Now, you are tasked to perform Incident Response to find useful evidence.

I.II - Goal

- Identify Email Subject
- Identify Document Name
- Identify Creation Time, Access Time, Modified Time
- Identify Sender Name
- Identify Launched Programs
- Listing Detection Method

I.III - Lab Setup

- VMware - Windows 7 x64
- Billing_Receipt.docx

I.IV - Tools Used

- FTK Imager Lite
- CodeBeautify.org
- Volatility 2.6
- Command Prompt
- Notepad ++

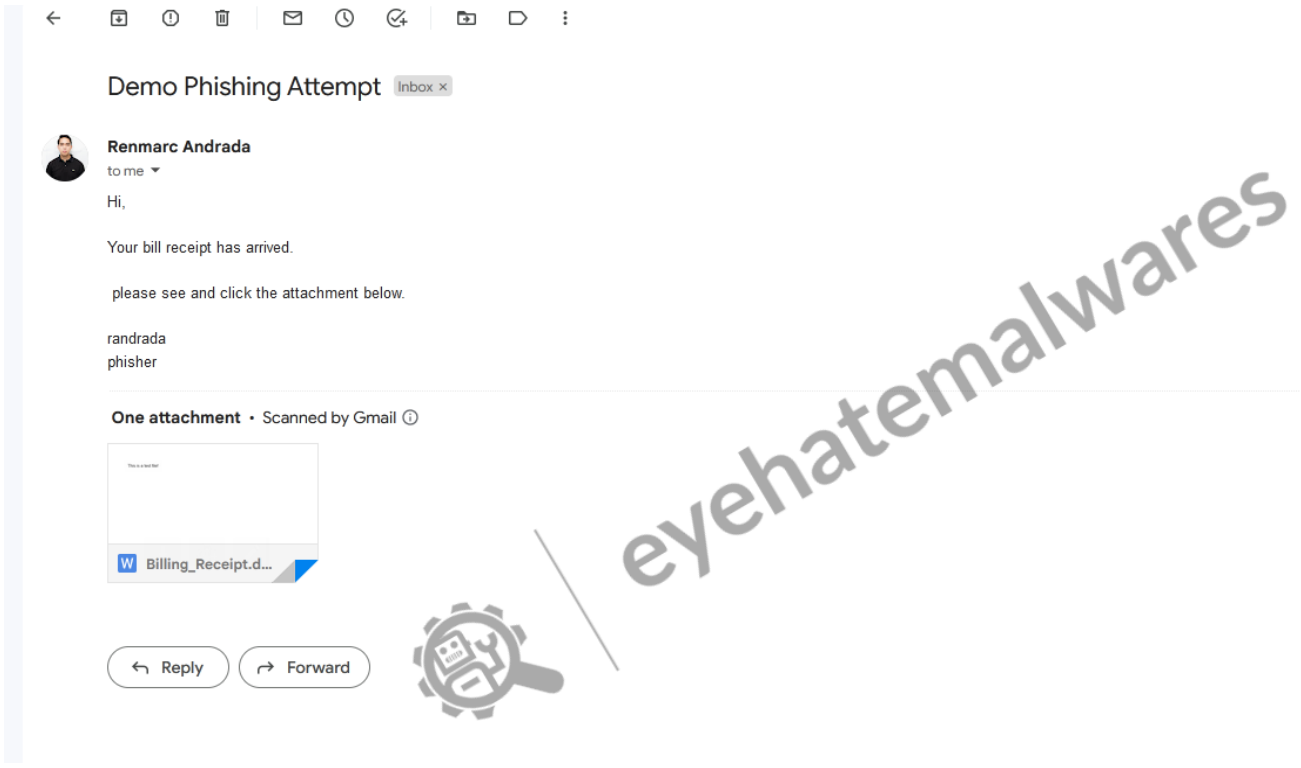

Figure I.I: Sample Phishing Email

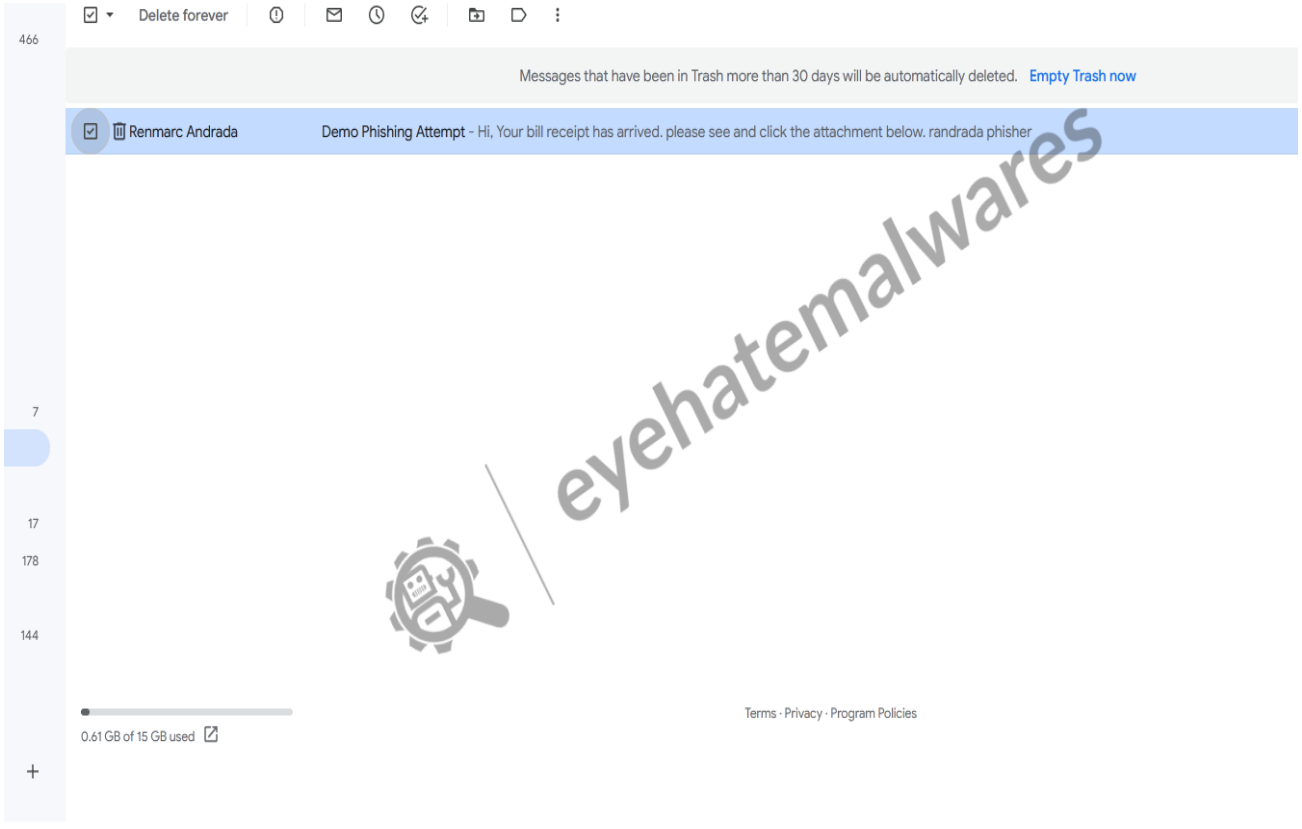

Figure I.II: Trashing Email

II – Memory Acquisition

Let's acquire the volatile memory of the suspected system.

In our case, we will use the tool **Access Data FTK Imager Lite**.

You can download and learn how the tool works here:
https://eyehatemalwares.com/digital-forensics/memory-acquisition/accessdata-ftk/)

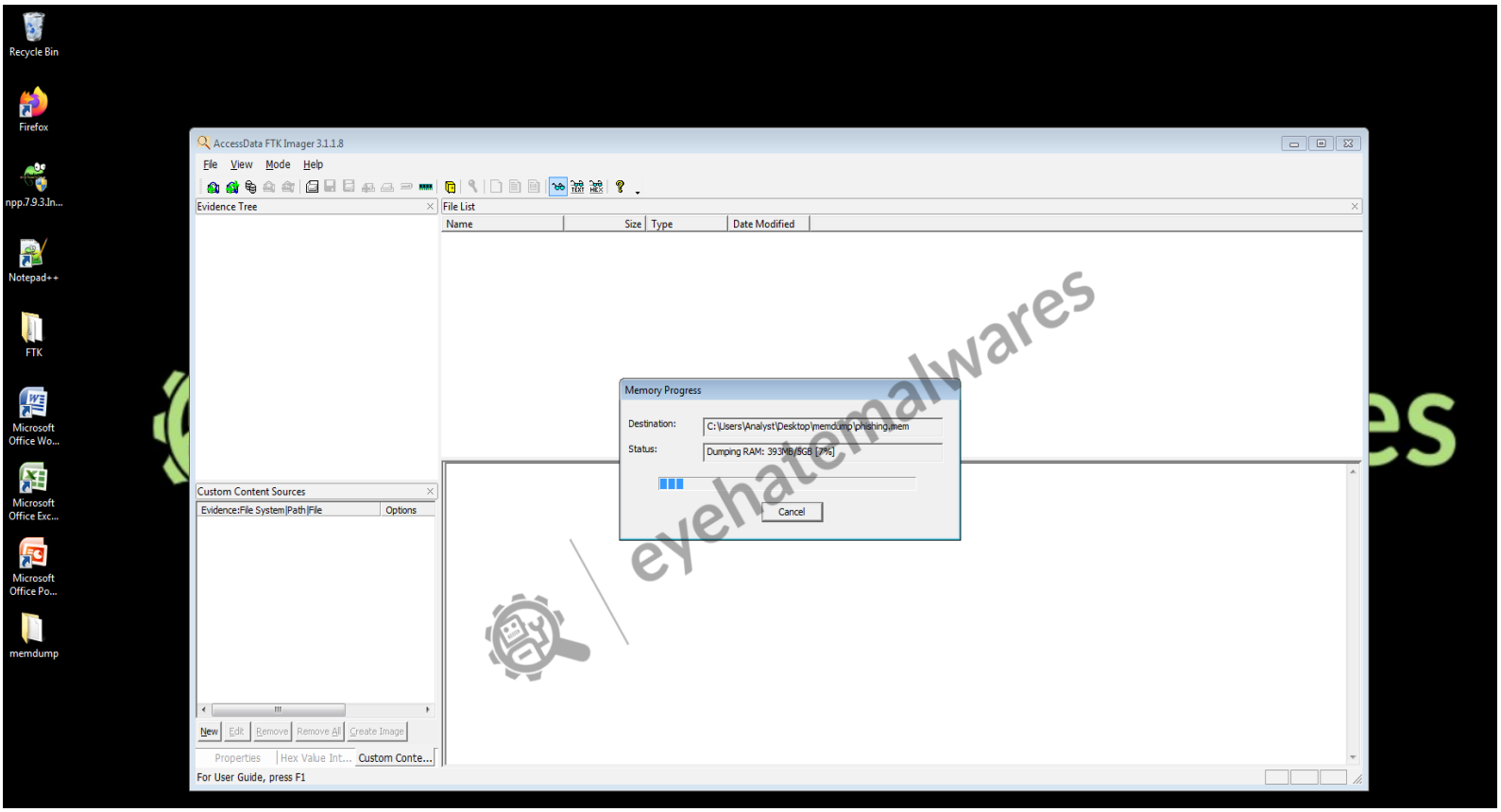

Figure II.I.I: Memory Acquisition via Access Data FTK Imager Lite

In the real world, after capturing the volatile image of the suspicious endpoint, use a hashing tool to compute the hash value of the acquired image. This is one of the right ways to handle collected evidence and safeguard the integrity of the file.

Link of the tools that can be used for hashing:

- **HashMyFiles:** https://eyehatemalwares.com/malware-analysis/static-analysis/hashmyfiles/
- **Hasher:** https://eyehatemalwares.com/incident-response/eztools/hasher/

III - Memory Analysis

Since we've already captured and hashed the image of the suspected endpoint we can now proceed to analysis stage.

In this section, we will discuss different approach in order to achieve our goal.

III.I – Approach A: Identify Document Name

> **Why File Handles?** In Windows systems, this is used to identify the file in many function calls. Each file handle and file object is generally unique to each process that opens a file – the only exceptions to this are when a file handle help by a process is duplicated, or when a child process inherits the file handles of the parent process. In other words, it is a temporary reference number that an operating system assigns to a file requested by a user to be opened.

By using Memory Analysis tool like Volatility we can track any open file handles that the user opened. But how? That's what we'll be discussing below.

First, we must identify if there are any process opened that is related to document opening or creation such as WINWORD.exe or EXCEL.exe.

To do this, we can use Volatility **pstree** and **handles** plugin.

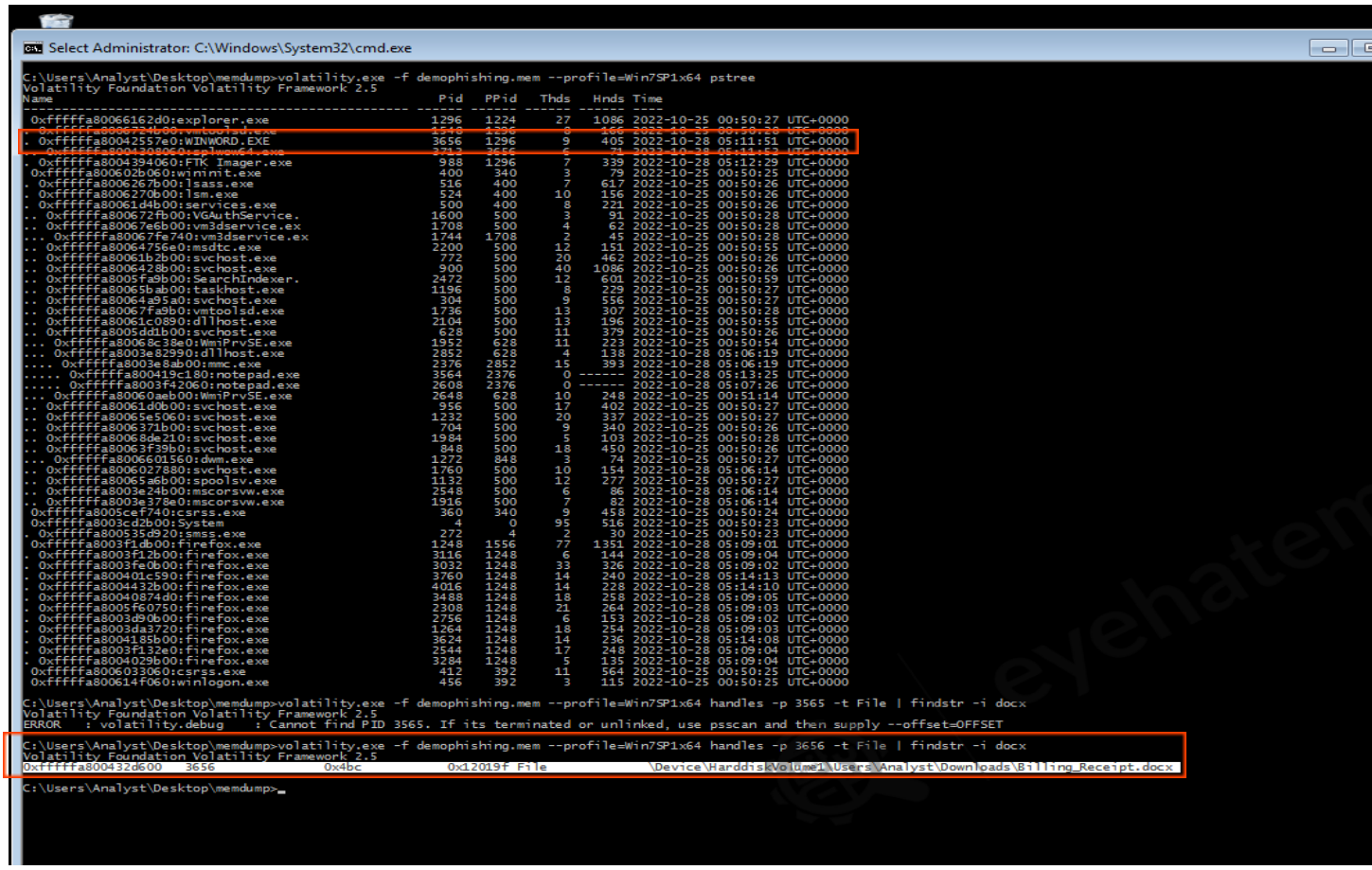

Figure III.I.I: Volatility pstree && handles plugin

In the figure above, we run **pstree** plugin to identify parent-child relationship of all the processes and we can see some interesting details such as:

- **Process Name/PID:** WINWORD.exe:3656
- **Timestamp:** 2022-10-28 05:11:51 UTC+0000.

Next, since we identified that WINWORD.exe process exists we can then use Volatility **handles** plugin to enumerate its associated file handles by using the command below:

Volatility.exe -f <sample.mem> --profile=Win7SP1x64 handles -p 3656 -t File | findstr -i docx

We pipe the output to **findstr** command to that search for result that match our pattern “docx”.

After successful execution, we can now see some interesting string like

- **Physical Offset Address:** 0xfffffa800432d600
- **Parent-PID:** 3656
- **Handle:** 0x4bc
- **Access:** 0x12019f
- **Type:** File
- **Details:** \Device\HarddiskVolume1\Users\Analyst\Downloads\Billing_Receipt.docx

Summary:

As incident responders, one of our objectives is to have a general understanding of the attachments downloaded. Using this method, we were able to determine the document's name, which is **Billing Receipt.docx**.

III.II – Approach B: Identify Creation Time, Access Time, Modified Time

Why Timestamps? Timestamps of a process or document generated may be valuable to us as responders since they will allow us to pinpoint exactly when the incident occurred. This allows us to find some evidence that will aid in our effort to reconstruct the incident's timeline.

Note: At first, don't assume what you notice since adversaries may employ tactics like timestamp modification to misled responders. In this case, correlation of different evidence, timestamps might be a good use.

To be able to perform this approach, we will dump the $MFT entries that is stored inside our volatile memory.

But first, let us have a quick introduction to $MFT.

Windows uses NTFS file system that contains a file called the *master file table* or $MFT. In Windows there is at least one entry in the $MFT for every file on an NTFS file system volume, including the $MFT itself.

All information about a file, including its size, time and date stamps, permissions, and data content, is stored either in $MFT entries, or in space outside the $MFT that is described by $MFT entries.

Now, we have understood what $MFT does.

To perform this approach, we will use Volatility mftparser to list and dump $mft entries.

Command:
Volatility.exe -f <sample.mem> --profile=Win7SP1x64 mftparser –output-file=mft.txt -D <target_directory>

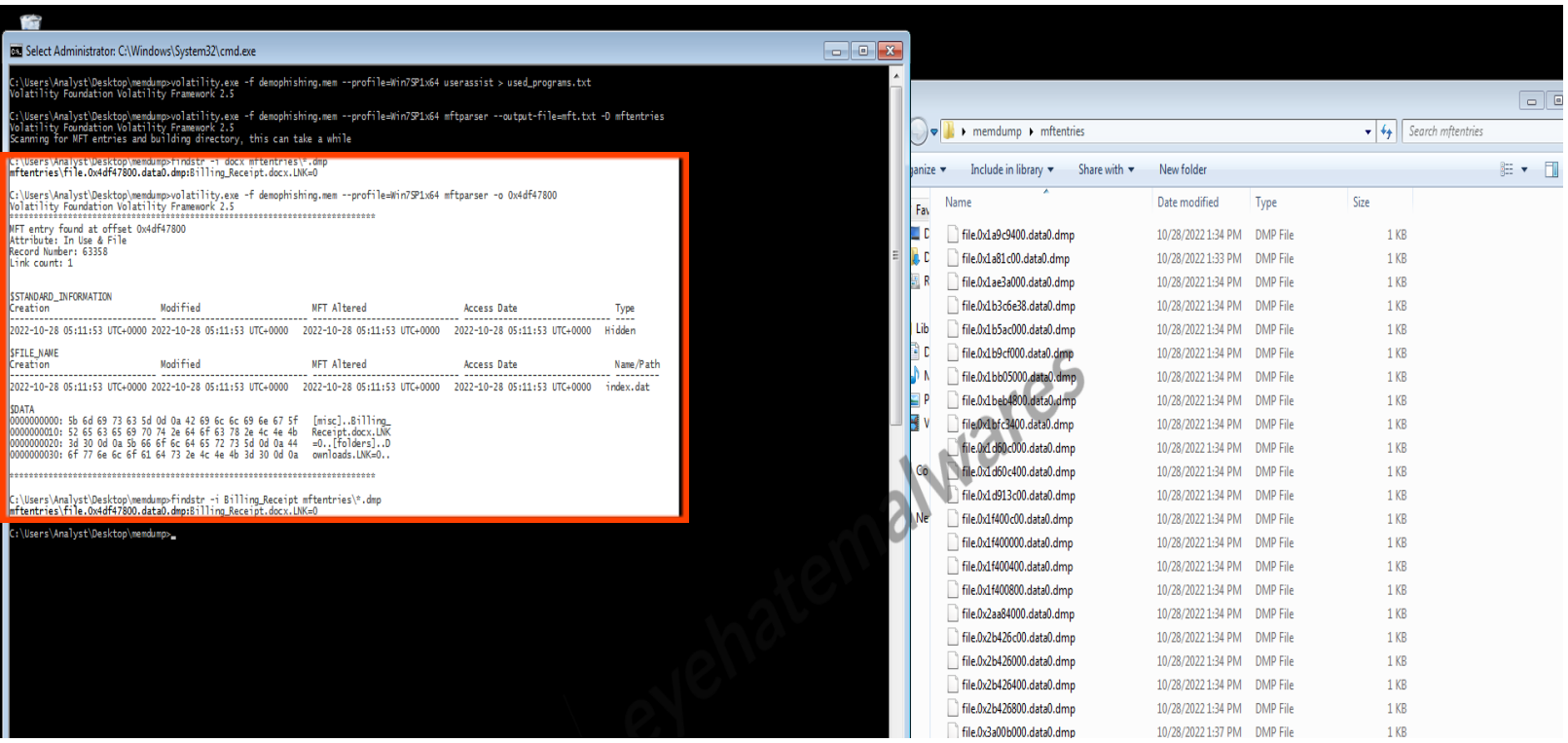

Figure III.II.I: Volatility mftparser

In Figure III.II, after we extracted all the $MFT entries we can then search for “docx” pattern by using findstr command: **findstr -i docx mftentries*.dmp**

In this case, our string search result is successful and gives us valuable detail such as offset address **0x4df47800.**

Then, we can use this address for the mftparser plugin to parse.
Command: **Volatility.exe -f <sample.mem> --profile=Win7SP1x64 mftparser –o 0x4df47800**

Now, we can see some valuable results such as:

- **MFT entry found at offset:** 0x4df47800
- **Attribute:** In Use & File
- **Record Number:** 63358
- **Link count:** 1
- **Creation Time:** 2022-10-28 05:11:53 UTC+0000
- **Modified:** 2022-10-28 05:11:53 UTC+0000
- **MFT Altered:** 2022-10-28 05:11:53 UTC+0000
- **Access Date:** 2022-10-28 05:11:53 UTC+0000
- **$DATA:** Billing_Receipt.docx.LNK

Summary:

When obtaining evidence, $MFT entries play a significant role, and as responders, we must maximize the value of this artifact. Using this method, we were able to determine the Timestamps associated with the Billing Receipt.docx and also give sufficient proof that the document was successfully executed by the user by seeing the **.lnk** extension after the document name.

III.III – Approach C: Identify Frequently Used Programs

Let us talk about Windows UserAssist

In Windows systems, there ways to identify what programs were recently executed on a system. This can be extremely valuable in an investigation where an examiner wishes to see if a particular application was run, such as an encryption or wiping tool. UserAssist data will include information on

whether an application was run from a shortcut(LNK file) or directly from the executable. This provides examiners with additional context around the execution of a program.

In a situation where we as responders aren't sure whether the document was executed by the user, we can check this artifact to see if there any instances of document creation/viewing tools are present.

Now, we will discuss how can we extract these artifacts from the captured image of our suspected endpoint.

Since we are diving into memory forensics, we will use Volatility's **userassist** plugin.

UserAssist Location:
NTUSER.DAT\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\

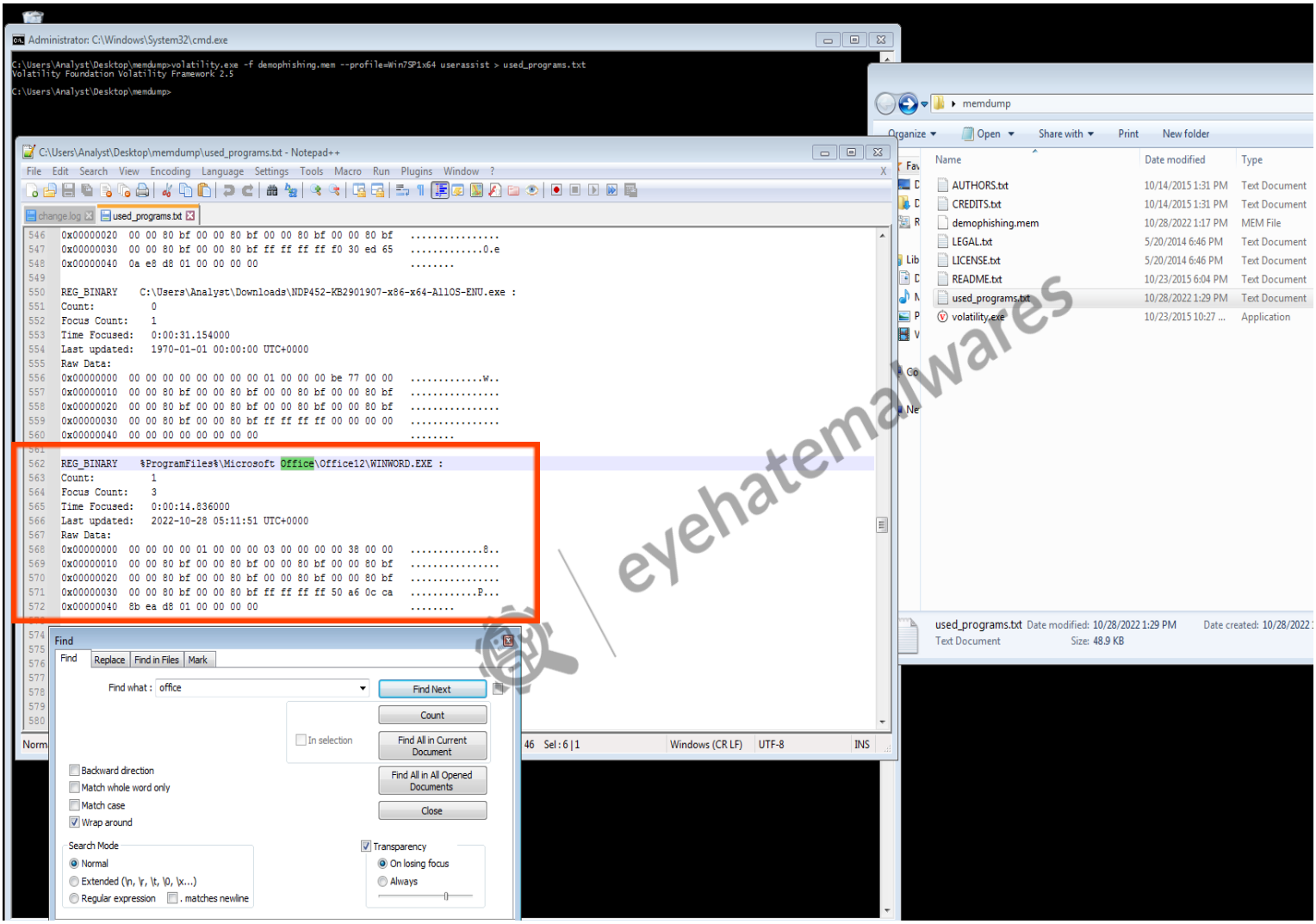

Figure III.III.I: Volatility userassist plugin

In Figure III.III.I, we use userassist plugin then saved the output as "user_programs.txt".

Command: volatility.exe -f <sample.mem> --profile=Win7SP1x64 userassist > used_programs.txt

After successful execution, we were able to gather some interesting strings such as:

- **Count**: 1
- **Focus Count:** 3

- **Time Focused:** 0:00:14.836000
- **Last Updated:** 2022-10-28 05:11:51 UTC+0000

Summary:

Using this approach, we were able to prove that the user recently run WINWORD.exe which associate that the user opened a document file.

Also, the timestamp from userassist can help us timeline the incident.

III.IV – Approach D: Identify Email Subject, Body and Sender Name

In Windows systems it uses *Windows Memory Management* to track all memory locations, whether the processes uses them or not. Specifies how much memory each process should be given. It decides which processes will be remembered and when. It also tracks when memory is released or when it is shared and changes the status accordingly. Windows uses memory management function called *paging* to store and retrieve data from a device's secondary storage to the primary storage.

One of the beauty of Windows Memory Management is it allows the data to be retrievable with the use of any memory forensic tools.

Note: Under some instances since Windows Memory is constantly changing it can overwrite the data that we've looking for.

In this approach, we will combine Volatility with another memory carving tool called *BulkExtractor.*

One way to perform our investigation effectively is we have to start with what we know, and since our goal is to find evidence of a possible phishing incident a web browser might a good start to perform our task.

To start, we will list all the running processes and identify the parent-child relationship and we can do that by using Volatility pstree plugin.

Command: volatility.exe -f <sample.mem> --profile=Win7SP1x64 pstree

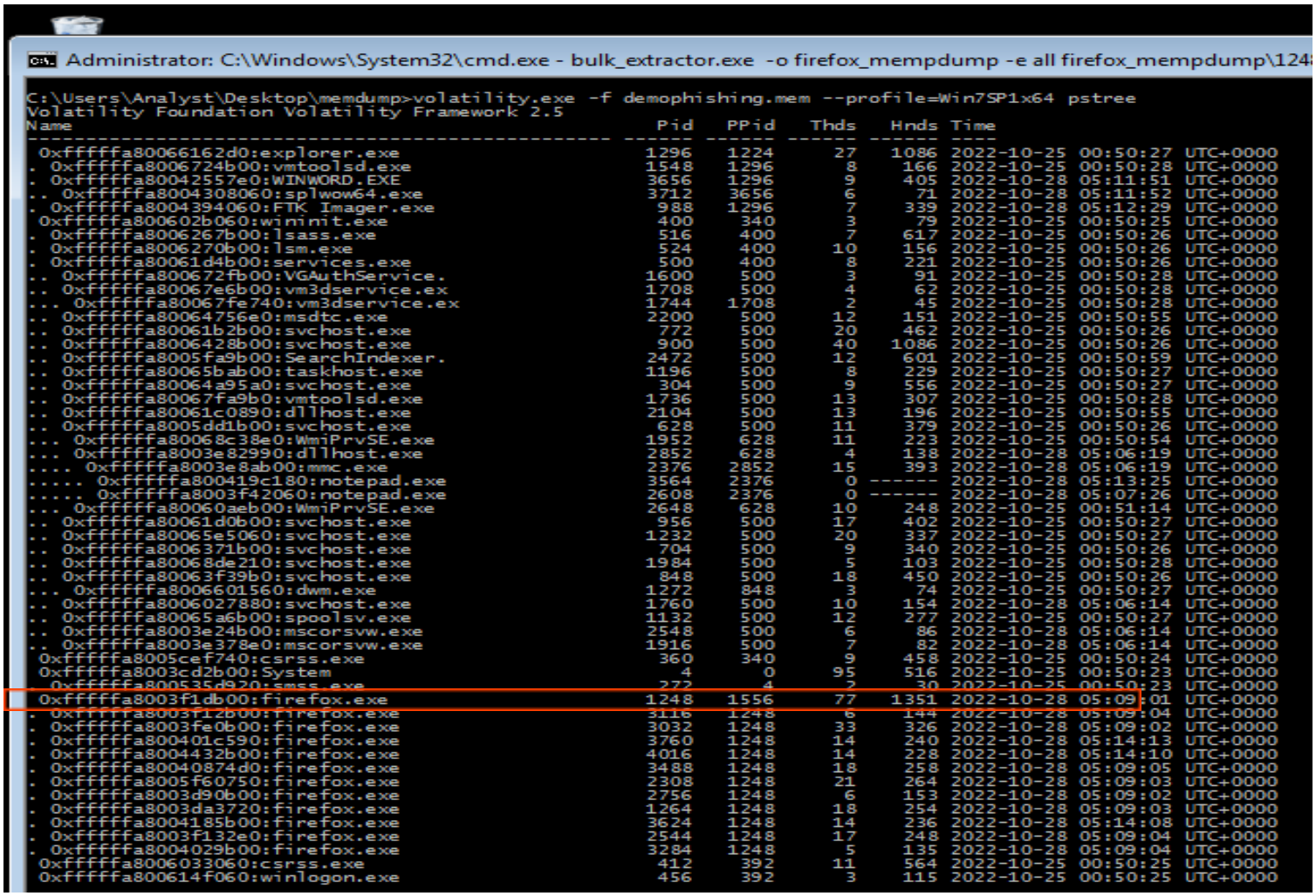

Figure III.IV.I: Volatility pstree plugin

After, successful execution we identify our browser process **Firefox.exe:1248**

Next, we will be using this details to perform process memory dump using Volatility **memdump** plugin.

Command: volatility.exe -f <sample.mem> --profile=Win7SP1x64 memdump -p 1248 -D <target>

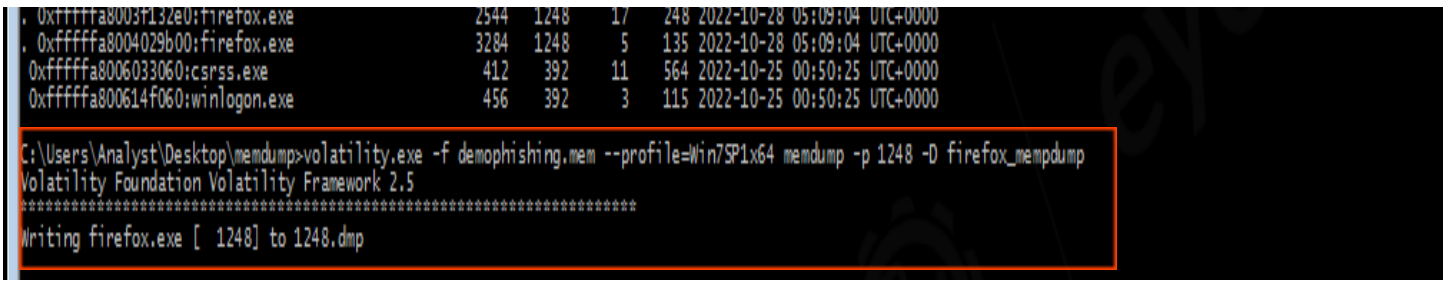

Figure III.IV.II: Volatility memdump plugin

After successful execution, **1248.dmp** will be saved to our target directory.

Now, we can use a tool *BulkExtractor* to extract information from this process dump.

To do this, we execute the following command.

Command: bulk_extractor.exe -o <target_dir> -e all <path_to_dmp_file>

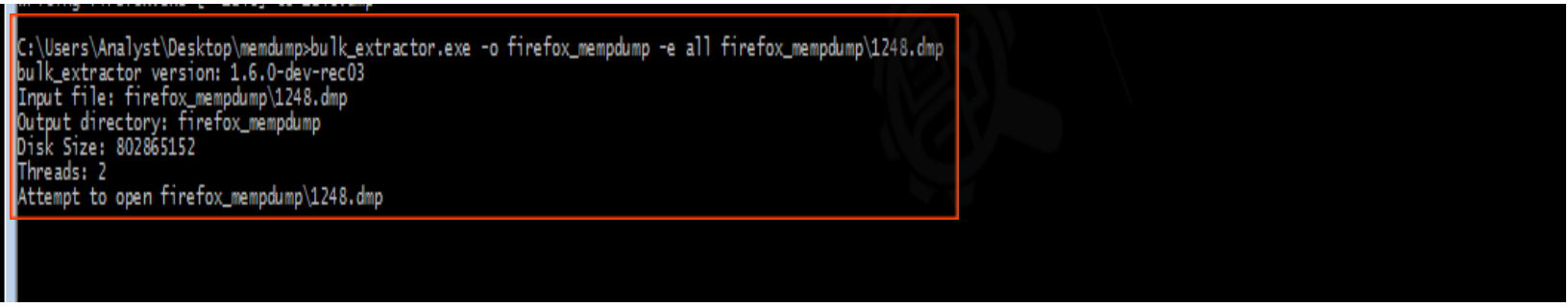

Figure III.IV.III: BulkExtractor

The **-e all** parameter perform all scan.

Now, this action may take a while.

After successful execution, a series of .txt files will be saved to the target directory.

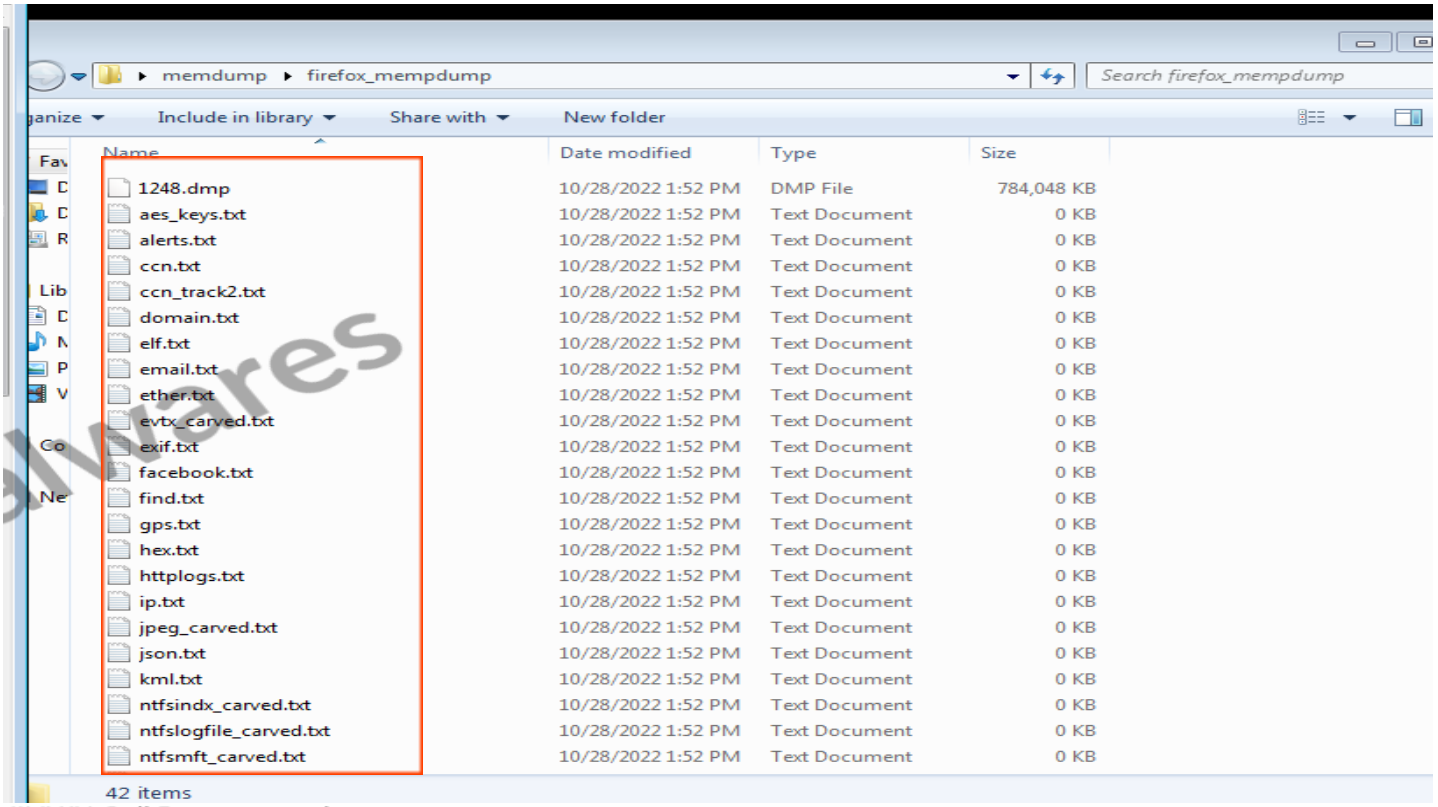

Figure III.IV.IV: BulkExtractor results

Next, we will perform string search and of course we will not perform this manually.

We will be using **findstr** to search for Billing_Receipt pattern and save the result as .txt file.

Command: findstr -i Billing_Receipt <dir>*.txt > DocxEvidence.txt

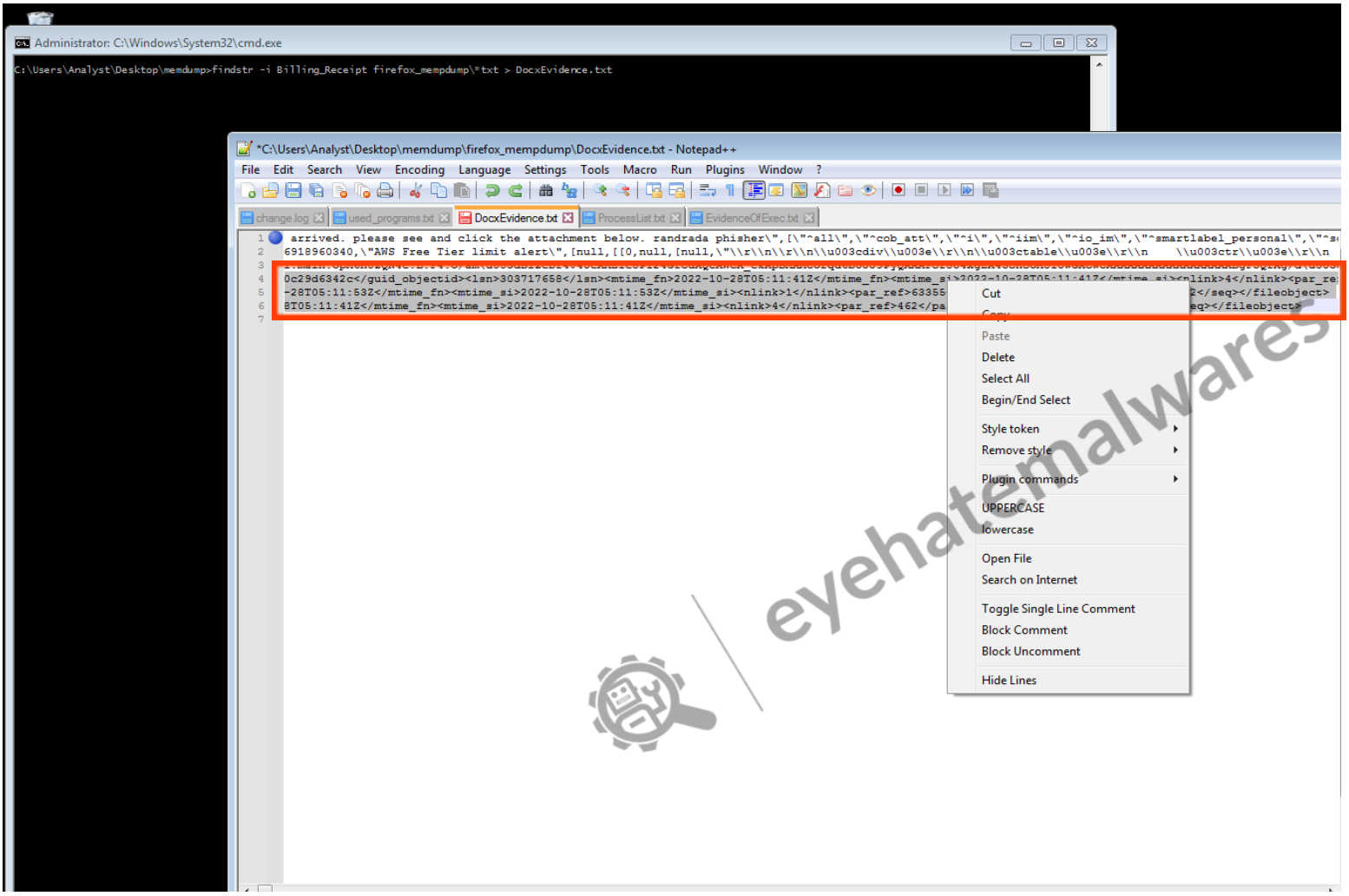

Figure III.IV.V: DocxEvidence.txt

Next, we can open this file in a text editor tool and we can see some interesting results.

Then, we can see this is an XML format.

Now, we can copy this XML formatted text to CodeBeautify.org to format it.

Figure III.IV.VI: Formatting XML format text

Now, we can see some interesting strings such as:

- **Billing_Receipt.docx Timestamps**
- **Billing_Receipt.lnk Timestamps**

This can be a valuable find to timeline the investigation.
Next, we will find email related evidence by searching “Billing” pattern to DocxEvidence.txt file.

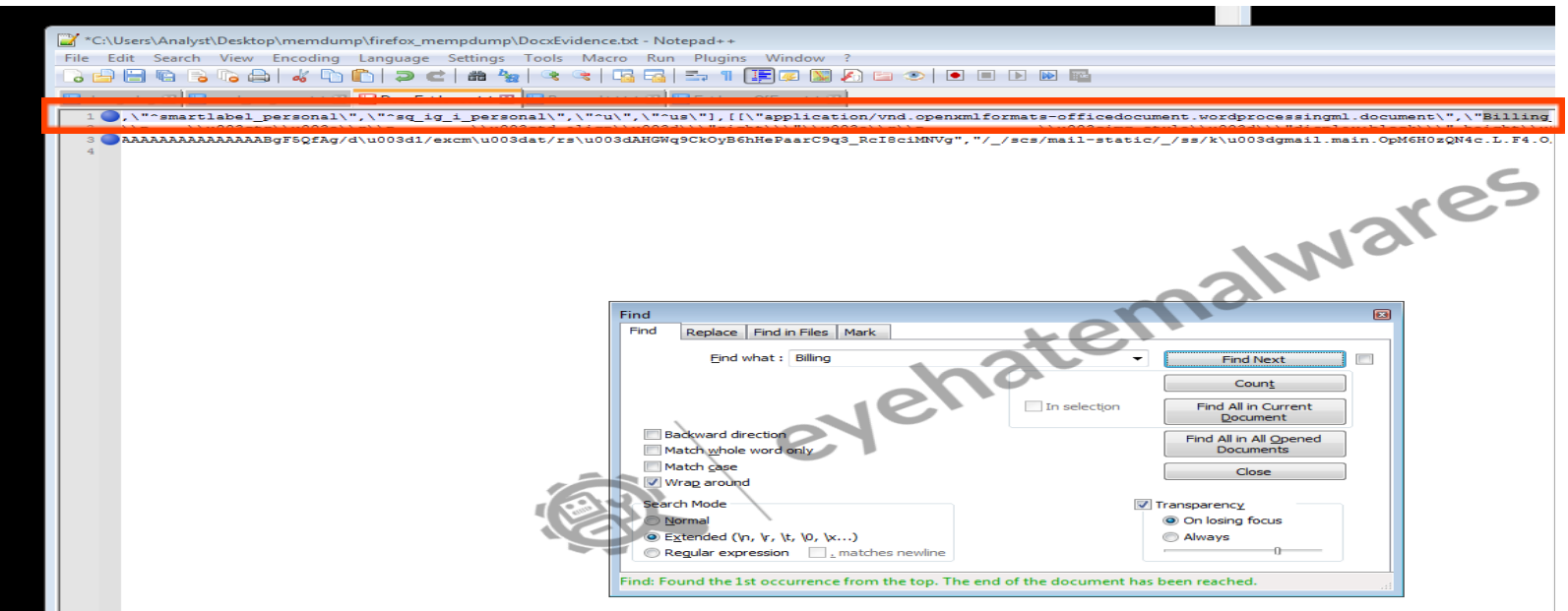

Figure III.IV.VII: Searching for Email Related Evidence

Our pattern is found at line 1 we eliminate the other lines. Since our text file is not readable, we can use the “replace” function and replace “,” with “\n” newline.

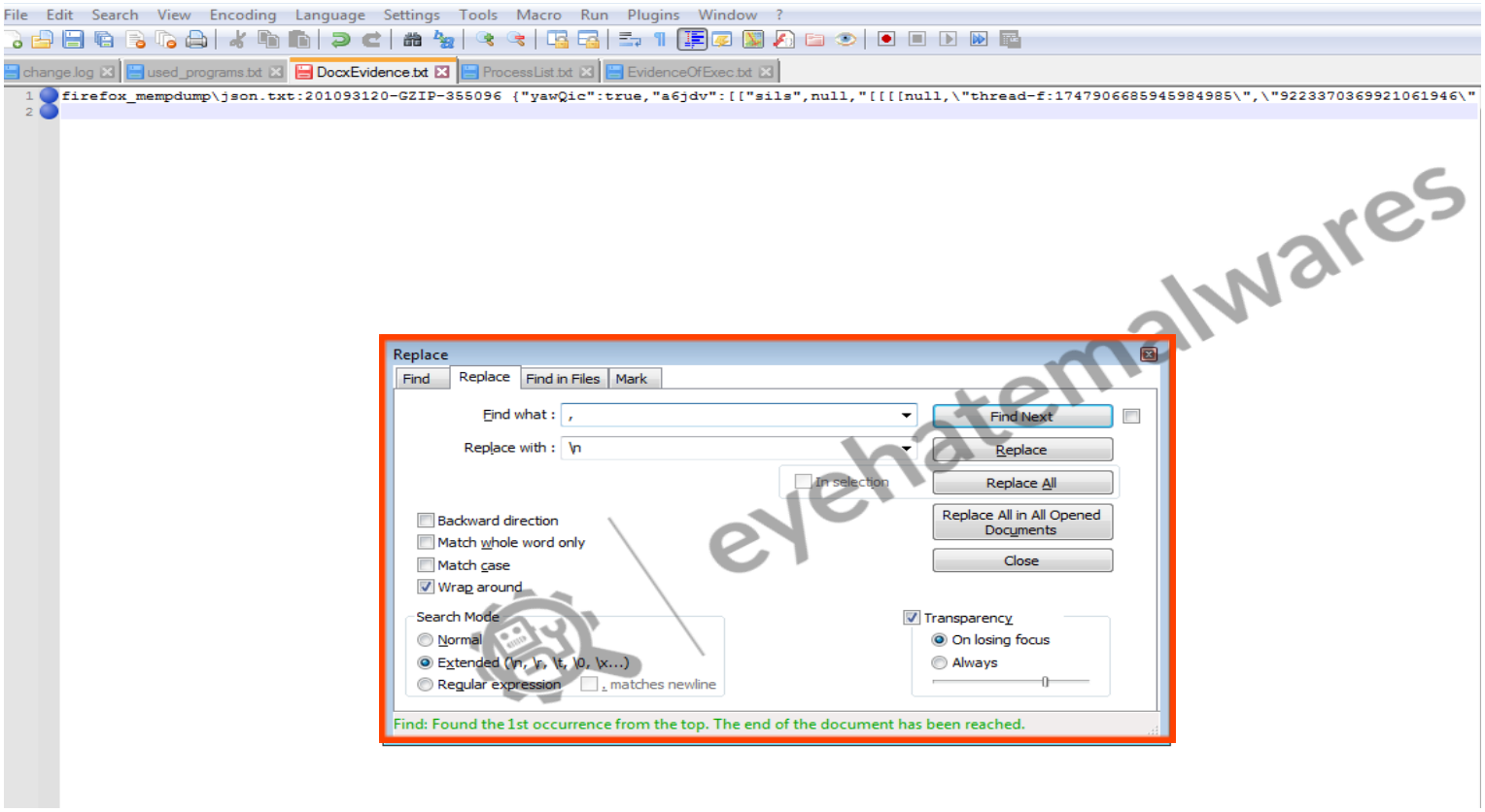

Figure III.IV.VIII: Replace function

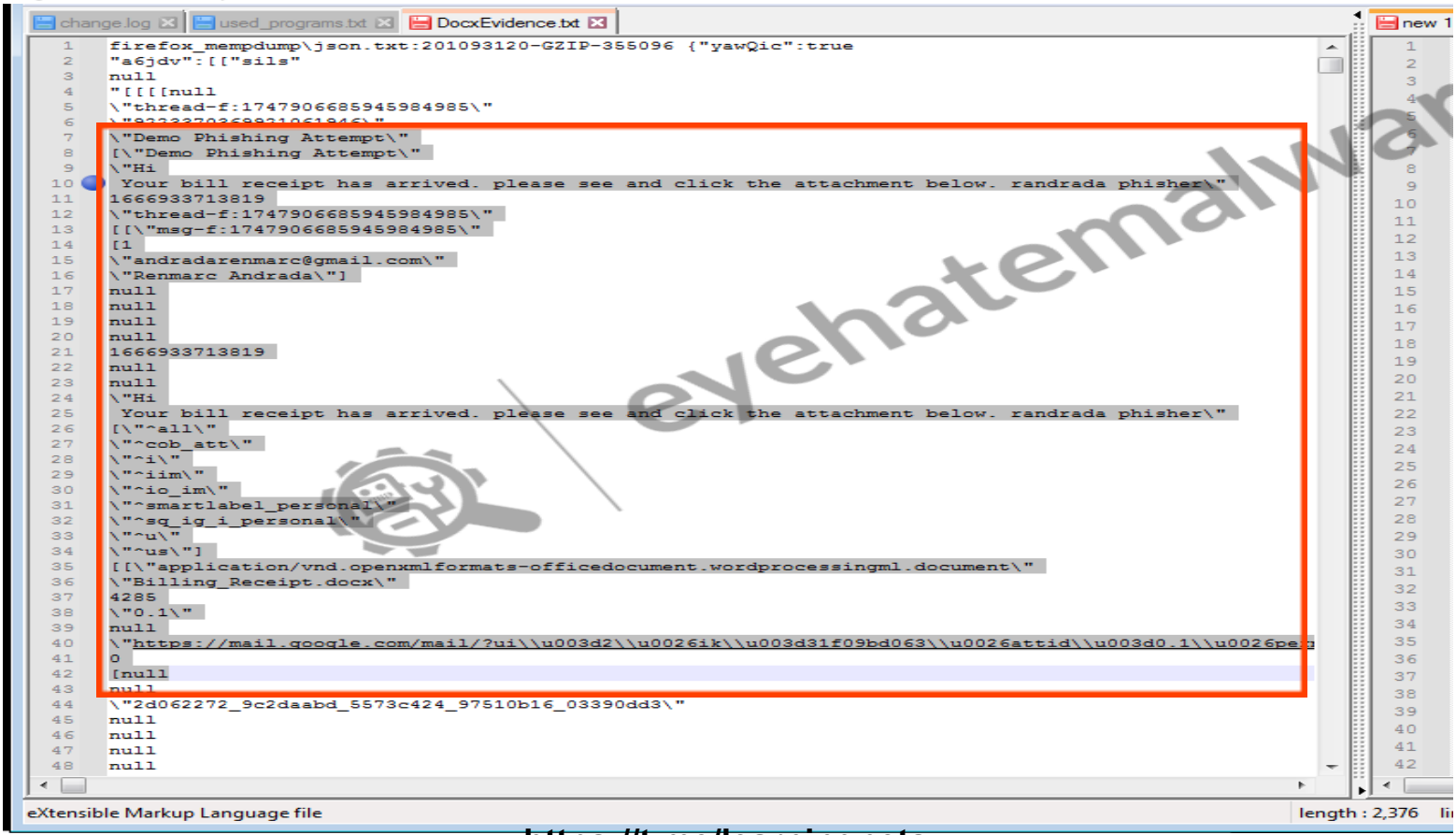

Figure III.IV.IX: Email Related Evidence

Now, it is readable and it can present us with some interesting strings such as:

Email Subject: Demo Phishing Attempt
Email Body:

“Hi Your bill has arrived. Please see and click the attachment below. randrada phisher”

Email Sender: andradarenmarc @gmail.com
Email Sender Name: Renmarc Andrada
Document Name: Billing_Receipt.docx

Summary:

Please keep in mind that Windows memory is constantly changing and may overwrite the data inside the process memory. This strategy may or may not work depending on when the problem was initially detected and the responder's response time.

IV – Endpoint Incident Response

Capturing the volatile image of the suspected endpoint is not always feasible, and doing memory capture on a critical system may be time-consuming depending on the endpoint's system resources.

In this situation, the responders can perform Live Forensics. This method may compromise the integrity of the suspected endpoint, but it is required when a memory capture is not possible.

In a phishing incident, there is an email with malicious attachment such as .docx, .iso and etc. files involved. These downloadable files contains malware or an exploit which able the adversaries to compromise the system, and before the initial compromise there might be changes first to the system such as:

- **Process Creation**
- **Process Injection**
- **Firewall Changes**
- **Creation of shortcut files or .lnk files**

In this section, we will talk about different approach on how to perform incident response list possible ways for detecting changes on the endpoint.

IV.I – Approach A: Identify Process Creation Timestamp

In Windows system, every file created or process that is being executed produces a timestamp which will provide us the data when was the file or process is created.

In performing incident response, this is a valuable data to look for and might help us understand and timeline the incident.

The image below is a .VBS script that uses WMI query to pull all the processes and its time creation, this allow us to see compare whether any process was created after the user launched the malicious document.

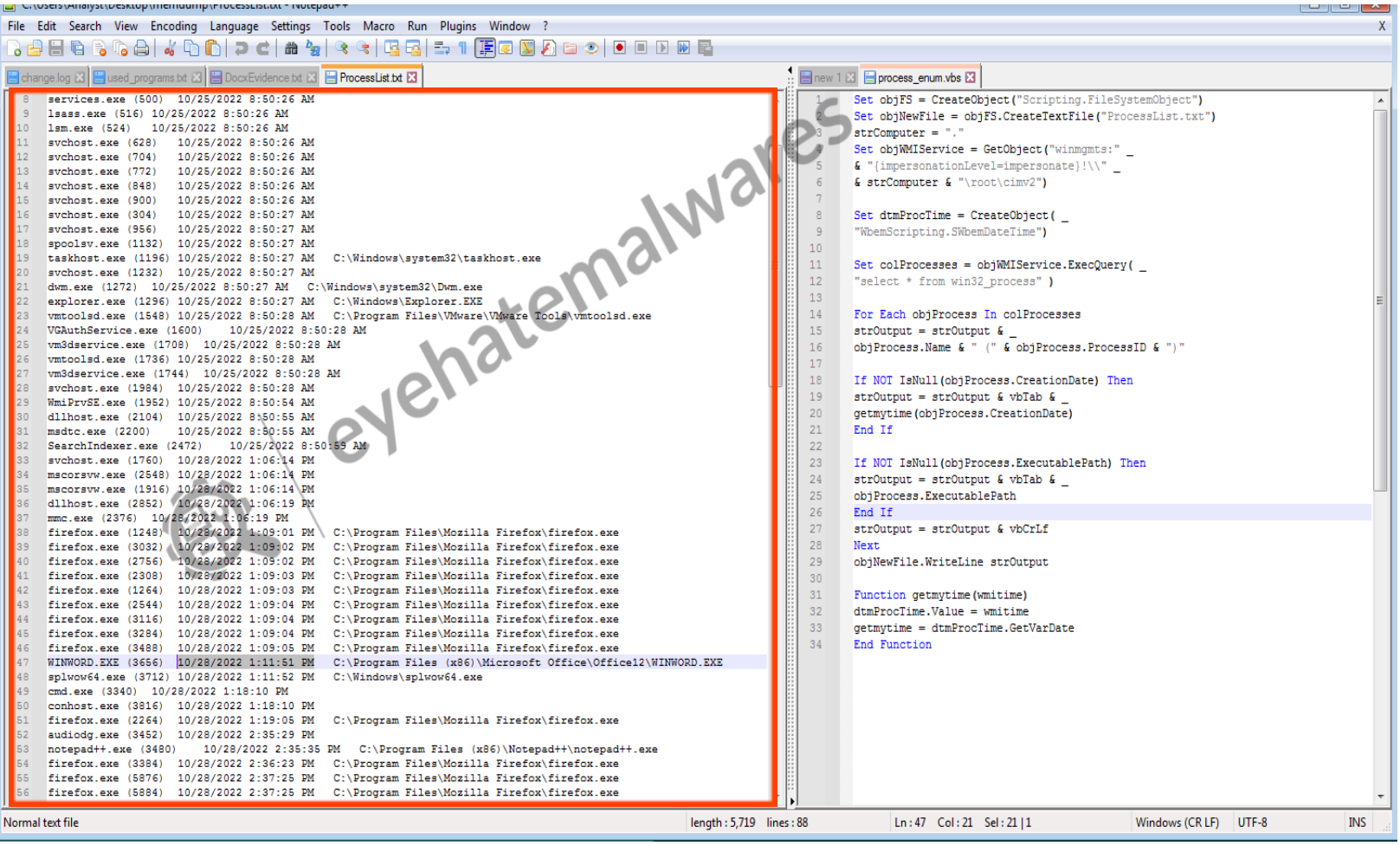

Figure IV.I.I: .VBS script using WMI query to pull processes time creation

In Figure IV.1, we see the script created a .txt file which contains the

- **Process Name**
- **Process ID**
- **Timestamp**
- **Executable Path**

Summary:

By identifying the time when was the process created, we can use this data to compare with other artifacts. Identifying whether a malicious process was launched right after the document was executed.

IV.II – Approach B: Windows Firewall Logs

Just like a bank robbery, the thief will try to contact its master for further instruction and malware will perform just like that. A malware right after the user launched the document is either a dropper which downloads an additional file or the malware that performs the malicious intention itself. But, in order to perform the connection to its command and control server there might be changes that will happen such as Firewall Rule Change.

In this case, looking inside the firewall log may help the responder to identify rule changes and identify odd IP addresses that are used to connect to the CNC server.

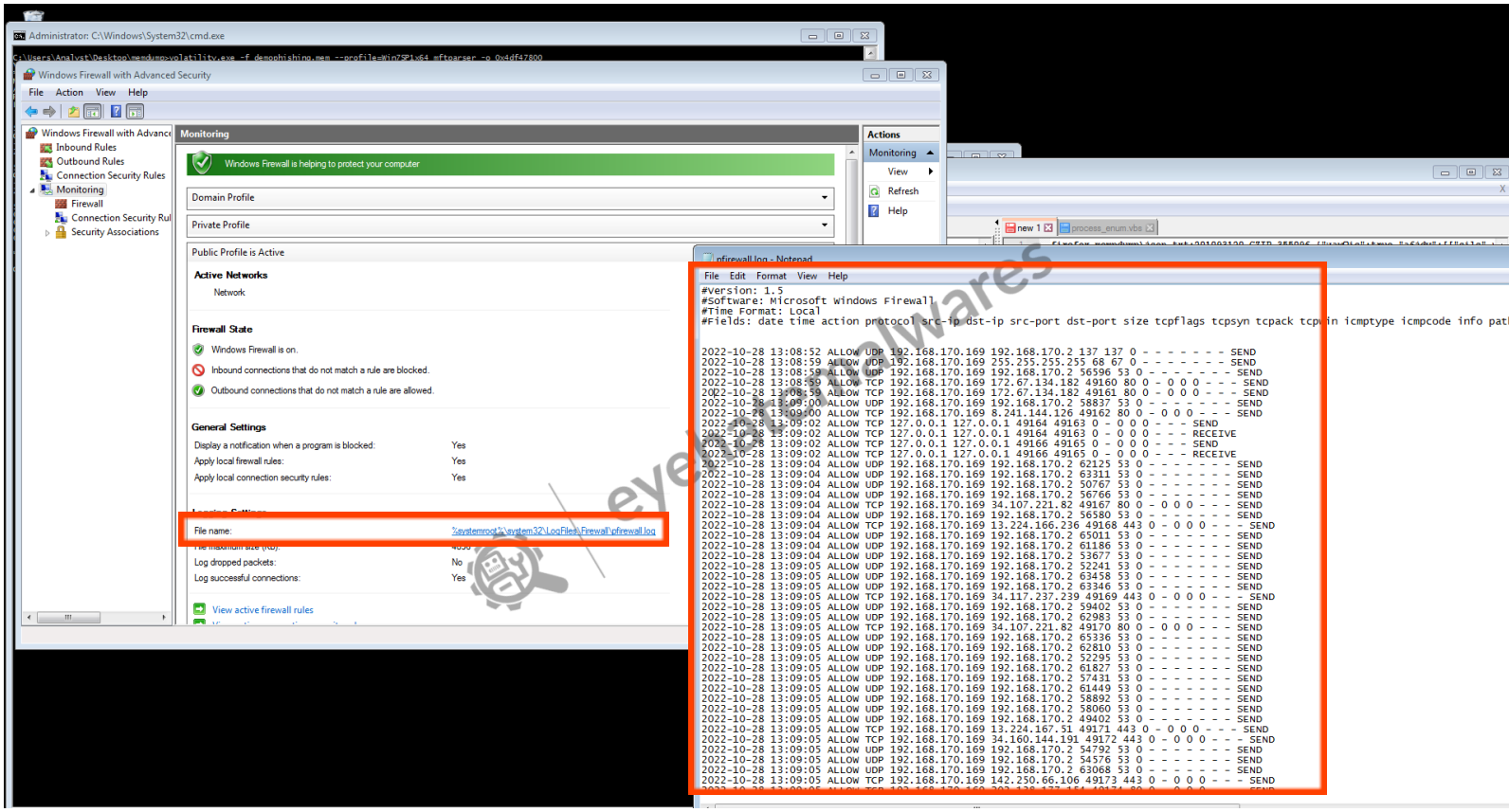

Figure IV.II.I: Firewall logs

In Figure IV.II.I, is an open firewall log.

We can navigate to the firewall logs through

Control Panel > System and Security\Windows Firewall > Advanced Settings > Monitoring > Logging Settings > File name

There are interesting strings that might interest us such as:

- **Connection Timestamp**
- **Rule**
- **IP Source**
- **IP Destination**
- **Port**
- **Status**

Summary:

Depending on your organization's Group Policy, endpoint's firewall logs should be enabled. This can contain information that can help the responder during investigation.

IV.III – Approach C: Shortcut Files or .LNK files

In the previous discussion, every file that is being executed will result to the creation of .lnk files and this artifact can be used as evidence that the file is executed. To locate .lnk files on Windows systems, the responder can navigate to:

C:\Users\<User>\AppData\Roaming\Microsoft\Windows\Recent

Here you can see all the .lnk file extensions which gives you an idea what files are executed on the system.

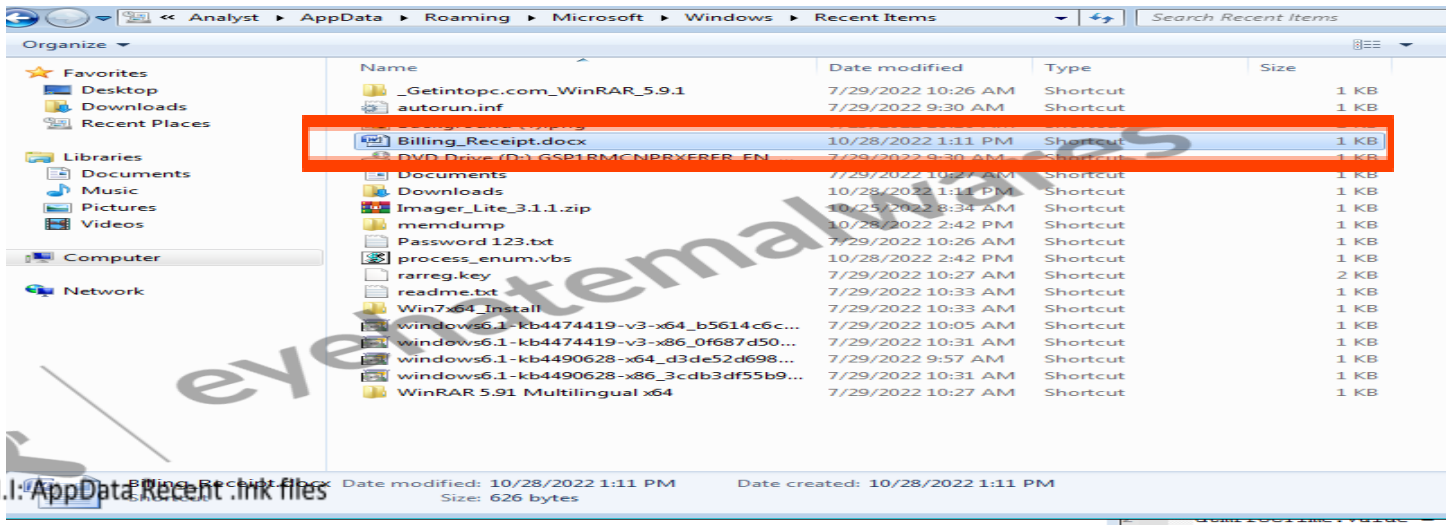

Figure IV.III.I: AppData Recent .lnk files

These files with .lnk extension can be use as Evidence of Execution. Also, there's a tool created by Eric Zimmerman called LECmd which parses .lnk files.

Learn and Download the tool LECmd here :
https://eyehatemalwares.com/incident-response/eztools/lecmd/

Different Endpoint Incident Response Approach can also be found here:
https://eyehatemalwares.com/incident-response/

V. References

Master File Table (Local File System)
https://learn microsoft.com/en-us/windows/win32/fileio/master-file-table

File Handles
https://learn.microsoft. com/en-us/windows/win32/fileio/file-handles
https://www.techopedia.com/definition/3313/file-handle

UserAssist
https://www.magnetforensics.com/blog/artifact-profile-userassist/

Windows Memory Management
https://www.geeksforgeeks.org/windows- memory-managment/

Volatility Foundation
https://github.com/volatilityfoundation/volatility/wiki/Command-Reference#memdump

Eyehatemalwares – EZ Tools
https://eyehatemalwares.com/incident-response/eztools-by-ezimmerman/

Eyehatemalwares – AccesData FTK imager
https://eyehatemalwares.com/digital-forensics/memory-acquisition/accessdata-ftk/

Eyehatemalwares - Memory Analysis using Volatility: mftparser
https://eyehatemalwares.com/digital-forensics/memory-analysis/volatility-mftparser/

Eyehatemalwares - Memory Analysis using Volatility: handles
https://eyehatemalwares.com/digital-forensics/memory-analysis/volatility-handles/

Eyehatemalwares - Memory Analysis using Volatility: moddump
https://eyehatemalwares.com/digital-forensics/memory-analysis/volatility-moddump/

Eyehatemalwares - Memory Analysis using Volatility: pstree
https://eyehatemalwares.com/digital-forensics/memory-analysis/volatility-pstree/

CodeBeautify.org
https://codebeautify.org/xmlviewer

List Running Processes and Their Creation Time
https://www.winhelponline.com/blog/list-running-processes-and-their-creation-times/