



# **TDX ARENA**

## **Certification Report**

**[student name]**

Final Assessment Report Submission

# Case: Imperial Memory

06/25/2025

## Executive Summary

Provide a brief overview of the case, key findings, the approach taken, and the outcome of the investigation.

The investigation began by examining the Emperor.vmem memory dump file to uncover sensitive data. Through analysis, a password was extracted which successfully decrypted the gift.7z archive. Inside this archive, a file named suspicious.docx was discovered, which appeared to be encrypted or obfuscated.

Upon further inspection using string analysis tools, it was determined that suspicious.docx contained embedded archive data. Using the same decompression tool employed on gift.7z, the .docx file was unzipped, revealing additional files located in a Desktop directory structure. Among these, a file named secret.txt contained clues suggesting that the final step involved hashing a specific file to retrieve the intended answer. This pointed toward a multi-stage analysis process involving memory forensics, archive extraction, file inspection, and hash-based verification.

---

## Findings and Analysis

### 1. Memory Dump Review (`Emperor.vmem`)

The investigation began by scanning the memory dump file `Emperor.vmem` using:

```
strings Emperor.vmem | grep "gift.7z"
```

This identified a reference to a protected archive (`gift.7z`) and revealed a password in plain text memory:

```
G6Vmc$Qd5cpM8ee#Ca=x&A3
```

This finding confirmed that sensitive data such as passwords can be recovered from memory using string analysis — a critical insight for forensic and incident response workflows.

---

### 2. Archive Extraction Using 7-Zip

The archive `gift.7z` was extracted using the 7-Zip command-line utility:

```
7z x gift.7z
```

With the recovered password, the archive was successfully decrypted, revealing a file named `suspicious.docx`. This document was initially assumed to be standard, but further analysis suggested otherwise.

---

### 3. Deep File Inspection

Inspection of `suspicious.docx` using string extraction tools indicated embedded archive content. As `.docx` files are essentially ZIP containers, it was unzipped with 7-Zip using the same approach:

```
7z x suspicious.docx
```

This action exposed an internal directory structure resembling a Desktop environment, containing multiple hidden files. Among them, `secret.txt` contained hints pointing toward a final challenge involving hashing.

## Tools and Technologies Used

### Tools:

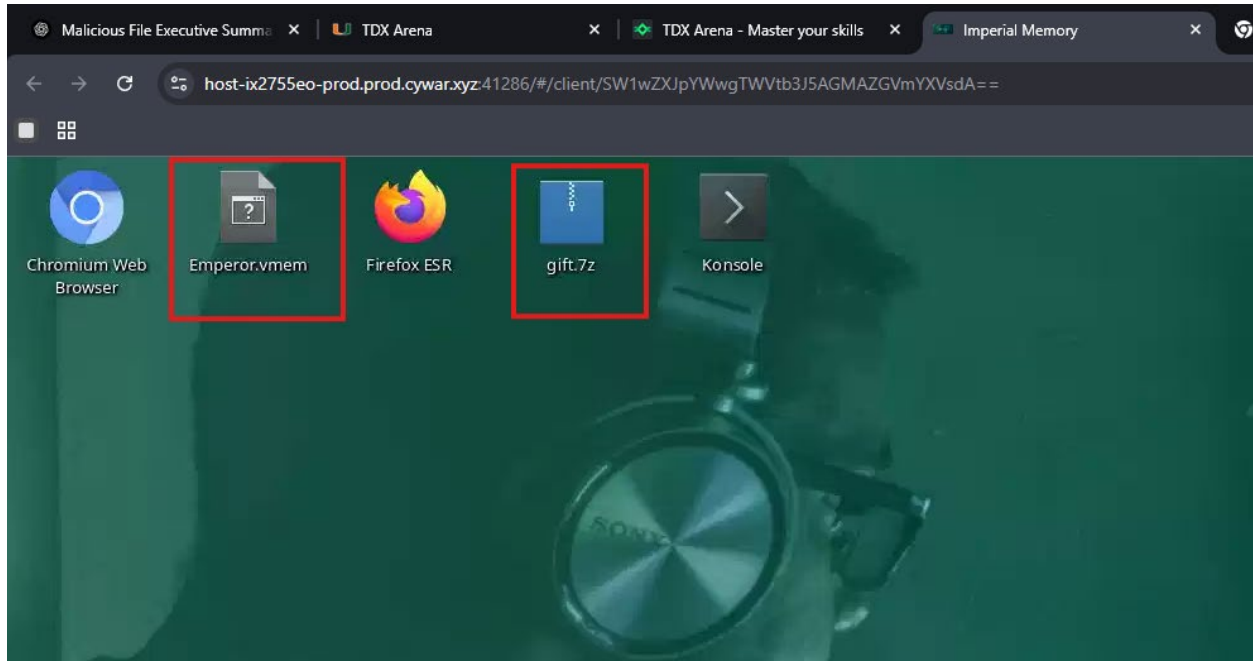
1. **strings (GNU Binutils)**
  - Used to extract readable text from the memory dump file (`Emperor.vmem`).
  - Helped identify references to the archive `gift.7z` and locate the password stored in memory.
2. **grep**
  - Used alongside `strings` to filter and quickly locate specific keywords such as `"gift.7z"` from a large volume of text output.
3. **7-Zip (7z command-line utility)**
  - Used to extract the contents of encrypted archives.
  - Successfully opened both `gift.7z` and `suspicious.docx`, revealing hidden files and data structures.
4. **Text viewers (e.g., `cat`, `less`, or graphical editors)**
  - Used to read the contents of files such as `secret.txt`, which contained clues or instructions for the final stage of analysis.
5. **Hashing utility (e.g., `sha256sum`, `md5sum`)**
  - While not explicitly performed in the previous steps, hashing was suggested as a necessary step in analyzing the final file, based on the contents of `secret.txt`.

---

### Technologies Involved:

1. **Memory Forensics**
  - Leveraged volatile memory analysis to uncover sensitive data like passwords and file references, demonstrating the risks of storing credentials in plaintext RAM.
2. **File Compression & Archiving**
  - Involved encrypted archive formats like `.7z` and `.docx` (a ZIP-based container).
  - Showed how data can be hidden or nested within multiple layers to evade detection.
3. **Command-line Interfaces (CLI)**
  - All major operations were conducted via terminal commands, showcasing the power and speed of CLI tools in forensic workflows.
4. **Data Obfuscation Techniques**
  - The use of a `.docx` file to hide additional archives and files reflects common tactics used in malware and digital forensics challenges to bypass casual inspection.

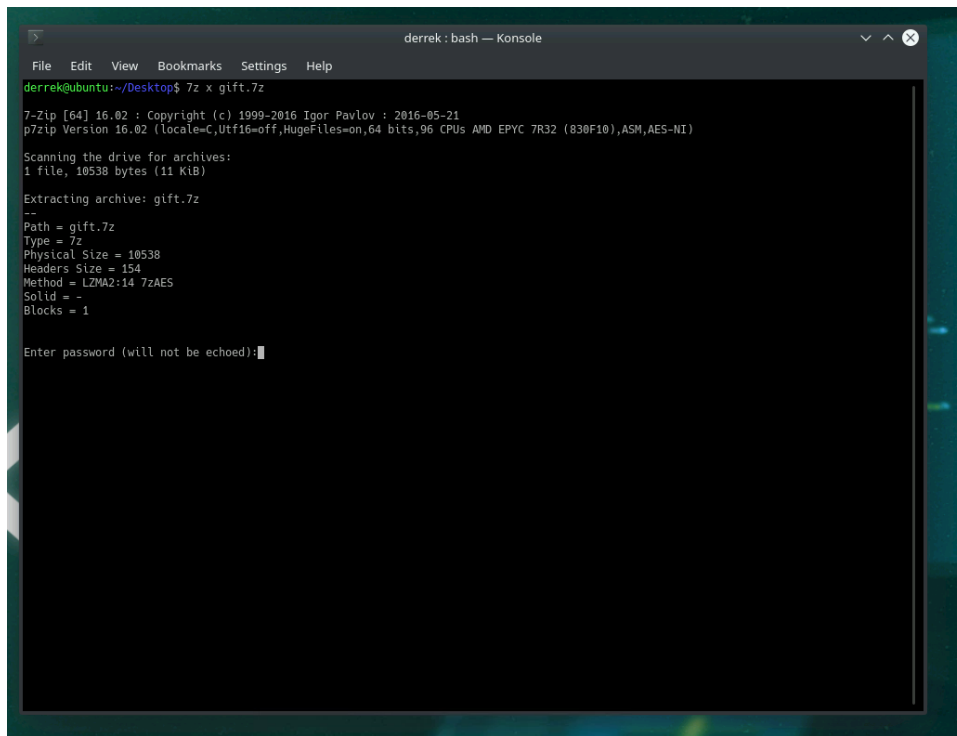
## Investigation Process



1 These are the files that needed to be investigated

[illegible]

2 This is the Information extracted form Emperor.vmem  
strings Emperor.vmem | grep "gift.7z"



A terminal window titled "derrek: bash — Konsole" showing the execution of the command `7z x gift.7z`. The output displays 7-Zip version 16.02 and system information. It then scans the drive for archives, finds `gift.7z`, and begins extraction. The file details shown are: Path = `gift.7z`, Type = `7z`, Physical Size = `10538`, Headers Size = `154`, Method = `LZMA2:14 7zAES`, Solid = `-`, and Blocks = `1`. The prompt "Enter password (will not be echoed):" is displayed at the bottom.

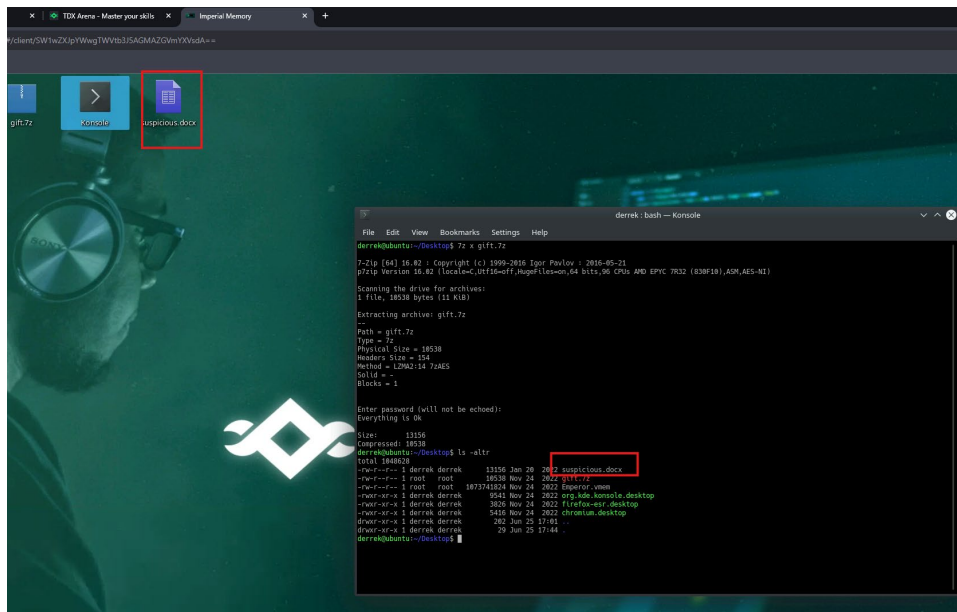
```
derrek@ubuntu:~/Desktop$ 7z x gift.7z
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=C,Utf16=off,HugeFiles=on,64 bits,96 CPUs AMD EPYC 7R32 (830F10),ASM,AES-NI)

Scanning the drive for archives:
1 file, 10538 bytes (11 KiB)

Extracting archive: gift.7z
--
Path = gift.7z
Type = 7z
Physical Size = 10538
Headers Size = 154
Method = LZMA2:14 7zAES
Solid = -
Blocks = 1

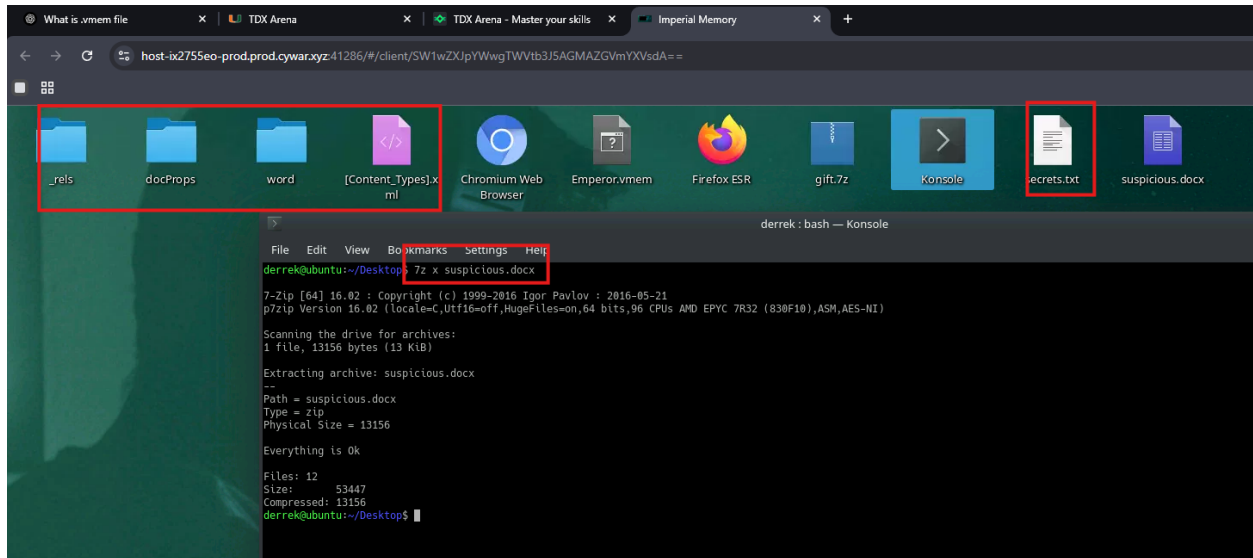
Enter password (will not be echoed):
```

3 7zip was used to unzip the gift.7z that needed the password extracted in step 2  
`7z x gift.7z`

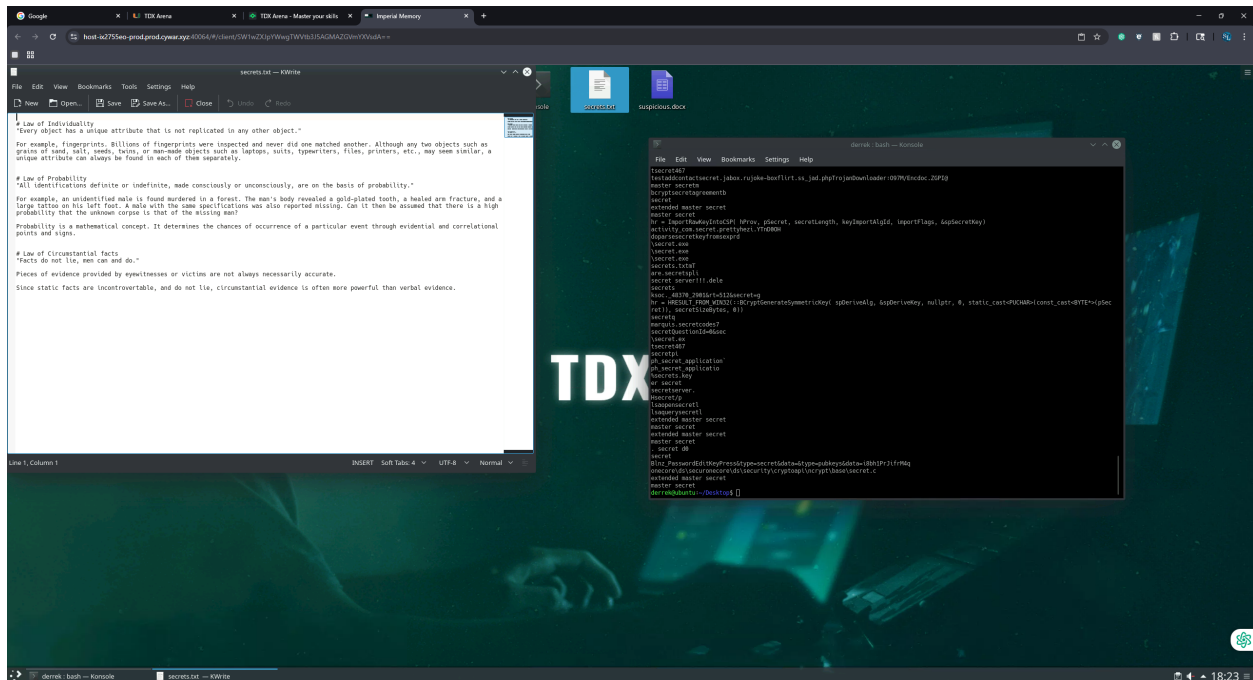


4 a suspicious.docx file was placed in the Desktop





6 after using 7z to decompress file, three folders and two files were placed in the Desktop



7 After reading secrets.txt it was implied that everything had a unique serial or code selecting this file provide a md5 hash by using

```
md5sum secrets.txt
```



---

## Recommendations

Based on the findings from the memory dump and file analysis, it is strongly recommended to implement stricter controls over sensitive data retention in memory. The fact that a password used to access an encrypted archive (`gift.7z`) was found in plain text within the `Emperor.vmem` file highlights the critical need for secure memory handling. Developers and system administrators should ensure that applications handling sensitive information clear memory buffers immediately after use. Additionally, endpoint protection tools should be configured to regularly scan volatile memory and flag the presence of plaintext credentials or unauthorized archive files.

From a security operations perspective, it is advisable to deploy automated tools that continuously monitor endpoints for suspicious file structures, such as encrypted `.docx` files containing hidden data or unusual archive nesting. Endpoint Detection and Response (EDR) systems, paired with scheduled memory analysis (e.g., via Volatility), can help detect and investigate threats that use advanced file obfuscation techniques. Lastly, hashing and integrity checks should be integrated into the incident response process to verify file tampering and to support forensic timelines during post-incident analysis.