Softwarica
COLLEGE OF IT & E-COMMERCE

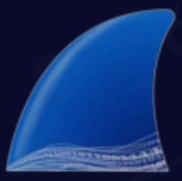
in collaboration with

Coventry University

ST4060CEM Digital Forensic Fundamentals
Ethical Hacking and Cyber Security

Memory Forensics

Email Forensics



Network Forensics

SUBMITTED BY
DINESH JOSHI
SUID: 240391
CUID: 15945644

SUBMITTED TO
RIKESH MAHARJAN
(Module Leader)

Acknowledgement

I want to really appreciate Mr. Rikesh Maharjan, my module leader, for his crucial advise, support, and assistance during the drafting of this report on the digital forensic investigation. His intelligent views and helpful critiques have greatly improved the level of my work and increased my knowledge of the field. I sincerely value the time and labor he invested into supporting me with this project.

Executive summary

This study investigates a Keed Loan Financial data breach that led to the disclosure of private customer information, including financial and personal data. The study tried to identify the attack vector, estimate the damage, and suggest remedies.

Evidence was gathered, analyzed, and reported using a six-phase forensic standard procedure. Programs such as FTK Imager, Volatility, and Ghidra were used to examine disk images, memory dumps, and network traces.

The breach started with a phishing email that contained obfuscated attachments, per the main findings. The exploitation of issues with FTP service configuration and LFI vulnerabilities came next. Memory forensics proved the existence of fileless malware activity.

The study offers an incident response plan and relates the attacker's strategies to the MITRE ATT&CK framework. Recommendations to prevent such incidents in the future include patch management, enhanced email security, endpoint protection, and regular memory analysis.

Table of Contents

List of Figure

1. Introduction

Keed Loan Financial, a well-known lender of both personal and business loans, experienced a **massive data breach** that allowed hackers to obtain access to sensitive customer information without consent. This assault exposed a variety of data, including bank account details, credit records, personally identifiable information (PII), and complete loan application files. The IT team looked for odd login patterns and questionable data access across many platforms after a number of customers reported fraudulent transactions. They discovered the trouble in this way.

Figure 1: Data Breach

In order to fully investigate this occurrence, Keed Loan Financial engaged a digital forensic investigator. Finding the attack routes, calculating the quantity of data lost, and creating practical solutions are the goals of the study. This paper presents the findings of a study on network traffic, memory, and email. Additionally discussed are the event reaction strategy and the digital forensic methods employed to investigate the events.

2. Task 1: Digital Forensic Methodology and Evidence Acquisition

2.1. Methodology Overview

The investigation involves a six-phase digital forensic method, which comprises preparation, identification, collection, inspection, analysis, and reporting. This extensive technique provides a systematic and repeatable process for handling digital evidence, keeping legal defensibility, and giving comprehensive coverage of all evidence sources. Each level builds on the one before it, producing an ordered structure that leads the investigator through the complexities of digital forensics.(Bluevoyant, 2022)

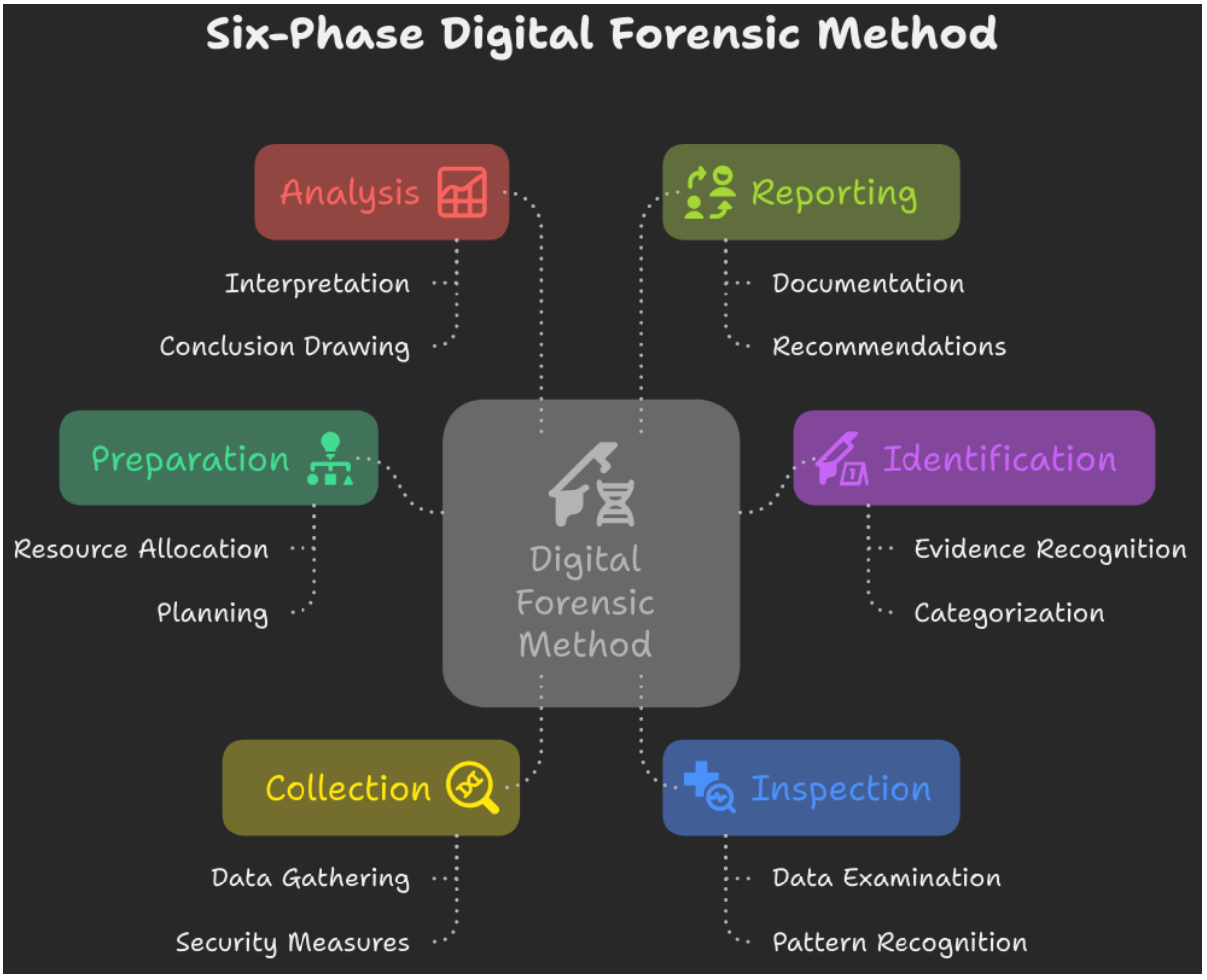


Figure 2: Digital Forensic Method

2.2. Phase 1: Preparation

Establishing the topic of the study is the first step in the planning process. Limiting illegal access to client data across various channels, such as endpoints, networks, and portable

devices, is part of this. A forensic team, which includes a lead investigator, network expert, memory analyst, and legal counsel, is then assembled to ensure a multidisciplinary approach to the inquiry. The necessary forensic tools are also acquired, including write-blockers, external drives, forensic workstations (Windows and Linux), Faraday bags, and evidence labels all of which are essential for the subsequent stages of the inquiry.

2.3. Phase 2: Identification

During the identification step, the primary focus is on inventorying the impacted assets. This means identifying and documenting all potentially hacked systems, including employee workstations, company email servers, loan processing servers, and any suspicious USB devices discovered during the initial investigation. Furthermore, there are two categories of evidence: non-volatile and volatile. Non-volatile evidence includes logs and disk data, whereas volatile evidence includes RAM, active programs, and network connections. This classification is critical for prioritizing the gathering efforts in the subsequent phase.

2.4. Phase 3: Collection

The true technique of gathering evidence makes the collection component vital. Disk imaging is done by creating sector-by-sector E01 and raw images of the impacted systems using programs like FTK Imager and dd. Through the prohibition of any adjustments during acquisition, hardware write-blockers are deployed to ensure the integrity of the source media. Applications like as Belkasoft Live RAM Capturer and Magnet RAM Capture are used to record dangerous memory from live hosts. A write-blocker and FTK Imager are also utilized to image a questionable USB device, correctly capturing its physical features for future use. Timeline reconstruction and threat analysis are further eased by the posting of network logs from firewalls, intrusion detection/prevention systems (IDS/IPS), and security information and event management (SIEM) platforms.

2.5. Phase 4: Examination

In the examination phase, the integrity of the collected data is verified through hash calculations, specifically MD5 and SHA-256, both pre- and post-imaging. This ensures that the data has not been altered during the collection process, with results recorded in a hash log for accountability. File system parsing is conducted using tools like Autopsy and Sleuth Kit to enumerate files, recover deleted artifacts, and extract metadata from the disk images. Log

parsing is also performed, utilizing Log2Timeline to correlate timestamps and convert Windows Event Logs into a more analyzable format, such as CSV.

2.6. Phase 5: Analysis

The analysis phase reconstructed the breach timeline through detailed examination of emails, network traffic, and volatile memory. Email forensics uncovered a multi-stage attack using corrupted attachments, hidden links, encoded messages, and obfuscation techniques to conceal attacker communications. Network forensics revealed initial access via a misconfigured FTP server allowing anonymous logins, exploitation of vulnerabilities like LFI and file upload bugs, successful reverse shell deployment, and malware downloads linked to known families such as Mirai and Amadey. Memory forensics identified suspicious running processes and extracted embedded flags through analysis with tools like Volatility and Ghidra, confirming active malicious code in memory not visible on disk or network logs. Together, these analyses provided a clear understanding of the attacker's methods, timeline, and tools.

2.7. Phase 6: Reporting

The final phase, reporting, involves the preparation of a formal report that documents the findings, methodologies, and recommendations derived from the investigation. This report is structured to provide clarity and comprehensiveness, ensuring that all aspects of the investigation are covered. It is also crucial to maintain a chain of custody throughout the investigation, ensuring that all evidence is properly labeled and documented to uphold its integrity and admissibility in any potential legal proceedings.

3. Task 2: Incident Response Plan Development

3.1. Preparation

The planning phase of the incident response plan underlines the need of having an updated asset inventory, which includes a complete description of all assets, network diagrams, and data classification policies. This inventory serves as a foundational factor for successful incident response. Additionally, adopting endpoint detection and response (EDR) solutions with real-time alerting features is critical for early detection of possible threats. Regular training sessions, including phishing simulations and security awareness programs for employees, are also crucial to guarantee that staff members are ready to spot and respond to security threats efficiently. (Paloalto, 2020)

3.2. Identification

In the identification process, monitoring Security Information and Event Management (SIEM) alerts becomes a priority. Alerts should be put up for abnormal logins, which may include geographical or time-based abnormalities, access escalations, and bulk file transfers. Furthermore, endpoint alerts should be enabled to detect strange processes, such as netcat or PowerShell scripts, which may suggest malicious activity. Tracking user reports through helpdesk tickets for unauthorized transaction complaints is also crucial, since it provides extra insights into potential security vulnerabilities.

3.3. Containment

Containment strategies are split into short-term and medium-term measures. In the short term, it is necessary to separate hacked workstations from the network and block any suspect accounts to avoid further unwanted access. Additionally, prohibiting malicious IP addresses and domains at the perimeter firewall is crucial to prevent risk. In the longer run, concerned subnets should be isolated, and access control lists (ACLs) should be installed to block access to services such as SMB and RDP. Implementing host-based firewall rules to limit outbound links improves containment efforts.

3.4. Eradication

The eradication step focuses on removing any malware present within the environment. This involves executing full-disk antivirus and anti-malware scans to discover and eradicate

unwanted software. Manual elimination of persistence artifacts, such as registry keys and scheduled tasks, is also critical to verify that the hazard is totally removed. Additionally, patching vulnerabilities is crucial; all operating systems, apps, and firmware must be kept up to date to shut any known vulnerabilities that may have been exploited during the attack. Resetting credentials for all affected users and mandating multi-factor authentication (MFA) across systems are also crucial tasks in this phase.

3.5. Recovery

In the recovery phase, restoring systems from known-good backups is a primary objective. This process should include certifying the integrity of the recovered systems through hash comparisons to ensure that no malicious code remains. Once systems are restored, they can be reconnected to the network in a controlled manner, with ongoing monitoring for any signs of re-infection. Validating security controls is also crucial during recovery; updating IDS/IPS signatures and reinforcing EDR policies will help to improve defenses against future assaults.

3.6. Post-Incident Activities

Post-event activities are crucial for boosting future incident response efforts. Conducting a lessons-learned review with stakeholders permits for the discovery of holes in the response process and provides a chance to adjust incident response playbooks accordingly. Updating policies is also crucial; enhancing email filtering, enforcing macro-blocking, and refining data exfiltration alerts can boost overall security posture. Finally, regulatory reporting must be conducted to warn data protection authorities and affected customers as required by GDPR and PCI-DSS regulations, guaranteeing compliance and transparency following the event.

Figure 3: Incident Response Steps

4. Task 3: Email System Investigation

4.1. Scope and Objectives

The investigation into Keed Loan Financial's data breach began with the discovery of a suspicious file named **keed_company.eml** in the corporate email directory. The primary objective was to determine whether email served as the initial attack vector or was used for data exfiltration. The analysis focused on examining suspicious emails, their headers, attachments, and any subsequent evidence that contributed to the breach.

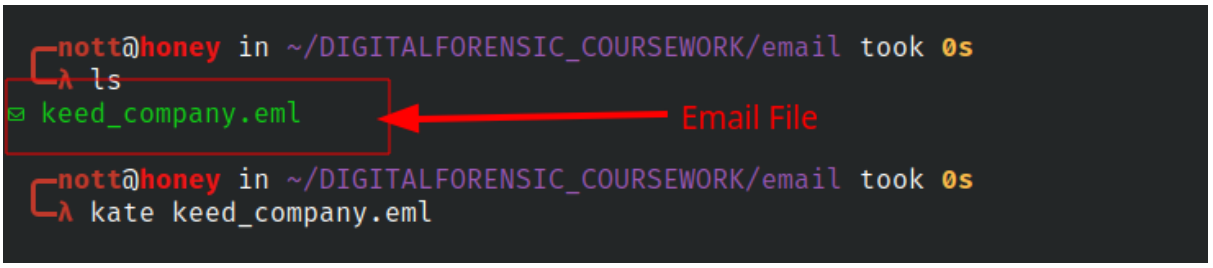


Figure 4: Email File

4.2. Email Header Analysis

Sending Address: shadow.reaper@darkmail.org

Recipient: security@keedcompany.com

Subject Line: "You Have 48 Hours to Find Me - Or Else"

Date: Tue, 1 Apr 2025 03:14:25 +0000

Message-Id: <7c6d3b8e.4f2a.5d9c.1a2b.3c8d9e8f0a3e@darkmail.org>

Content-Type and Encoding: text/plain; charset="utf-8

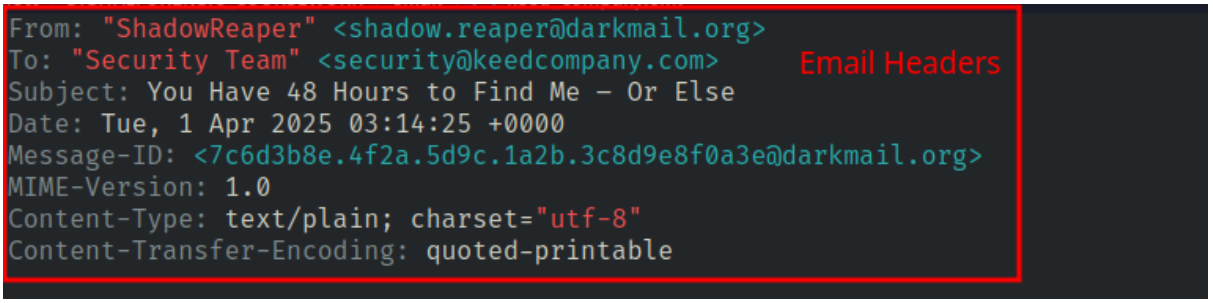


Figure 5: Email Header

Upon reviewing the header of the suspicious email, it was observed that the message originated from **shadow.reaper@darkmail.org** and was addressed to the security team at Keed Loan Financial. The **subject line**, "You Have 48 Hours to Find Me - Or Else," was both threatening and urgent, suggesting extortion tactics. The message was sent at an unusual hour, potentially to avoid detection. Notably, internal Exchange authentication headers were present on an externally sent email, raising concerns about possible spoofing, account compromise, or manipulation of message routing. The presence of an attachment was also confirmed, further elevating suspicion.

4.3. Attachment Analysis

The email contained a base64-encoded attachment. After decoding, it was found to be a PDF file; however, the document was malformed or corrupted. A detailed examination revealed that the PDF did not conform to the standard structure, which requires a header, body, cross-reference table (xref), and trailer.

Figure 6: base64 encoded data

Further analysis identified a recurring pattern—specifically, the insertion of the string 'aaa' at three locations within the file—which was responsible for the corruption. Once these extraneous patterns were removed, the PDF could be successfully downloaded and viewed, indicating the file had been intentionally obfuscated to evade detection or hinder analysis.

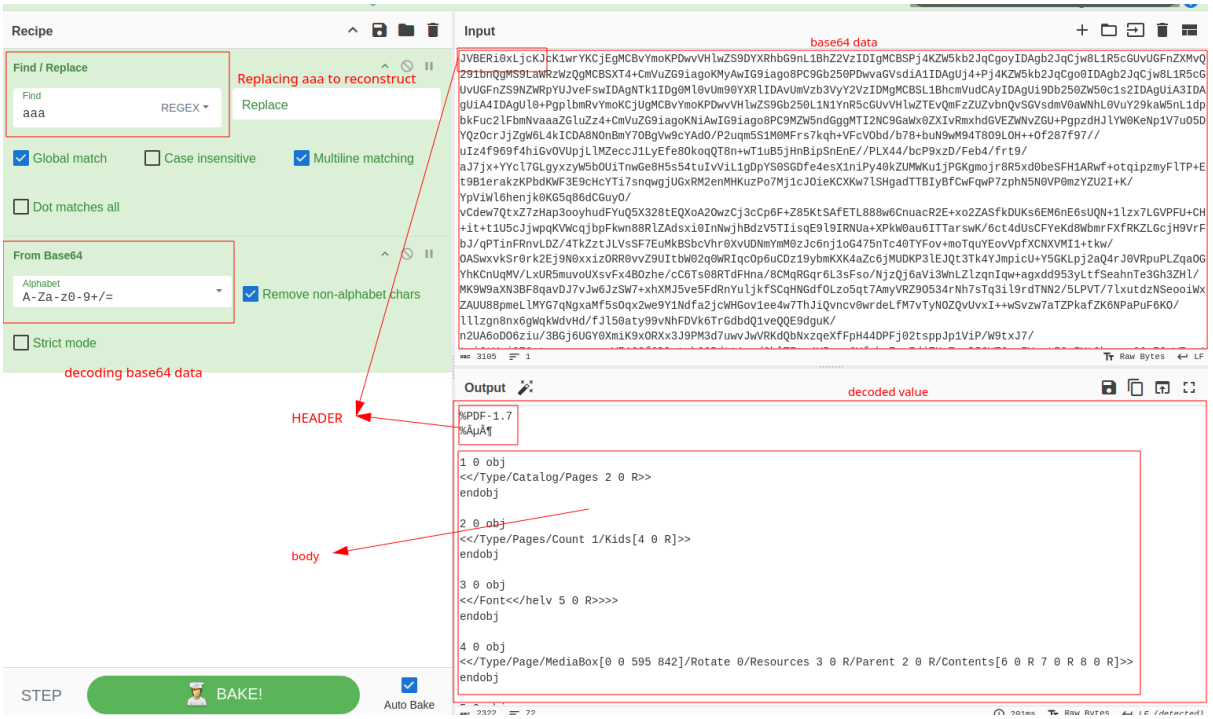


Figure 7: base64 to pdf

4.4. Discovery of Hidden Evidence

Upon reviewing the cleaned PDF, a hidden Google Drive link was discovered. This link led to the download of another email file, providing additional evidence and extending the chain of investigation. This step demonstrated the attacker's use of multi-stage delivery and hidden content to complicate forensic analysis.

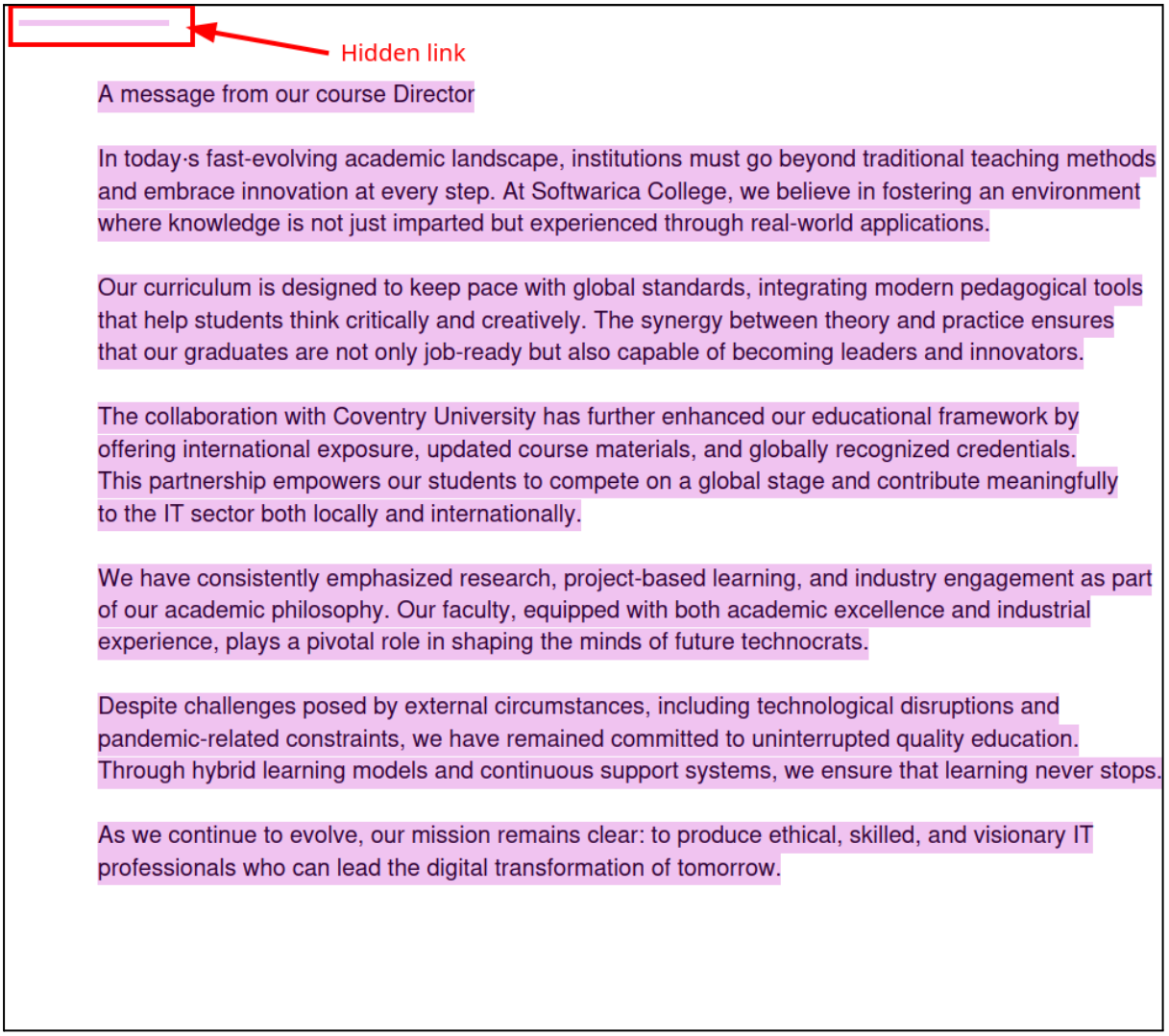


Figure 8: Hidden google drive link

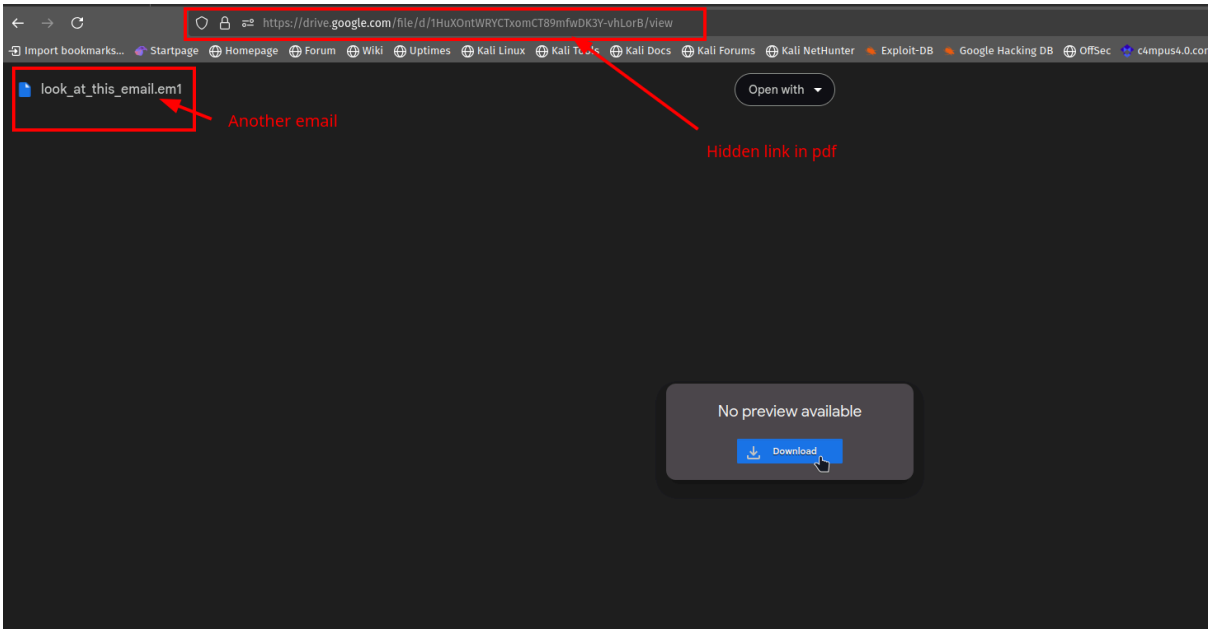


Figure 9: Another Email

Downloading this email leads to finding another email evidence.

4.5. Analysis of the Second Email

Sending Address: rikehmhj9@gmail.com

Recipient: keedhacker7@gmail.com

Subject Line: “look at this movie and get a flag”

Date: Wed, 4 Jun 2025 09:51:09 +0545

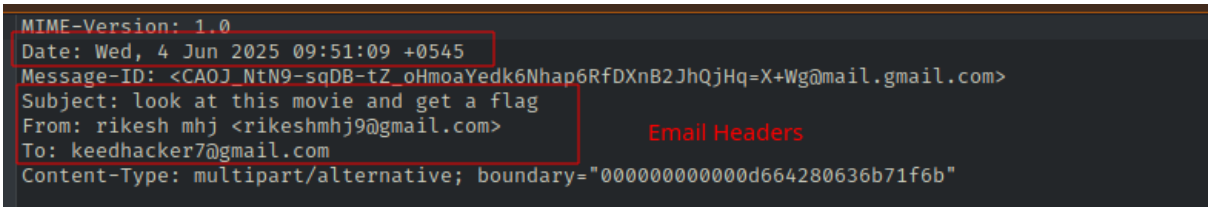


Figure 10: Second email Header

The second email, obtained from the Google Drive link, originated from **rikehmhj9@gmail.com** and was sent to **keedhacker7@gmail.com** with the subject “look at this movie and get a flag.”After decoding, the attachment was identified as a video file, which became the focus of further analysis.

The email included another base64-encoded attachment.

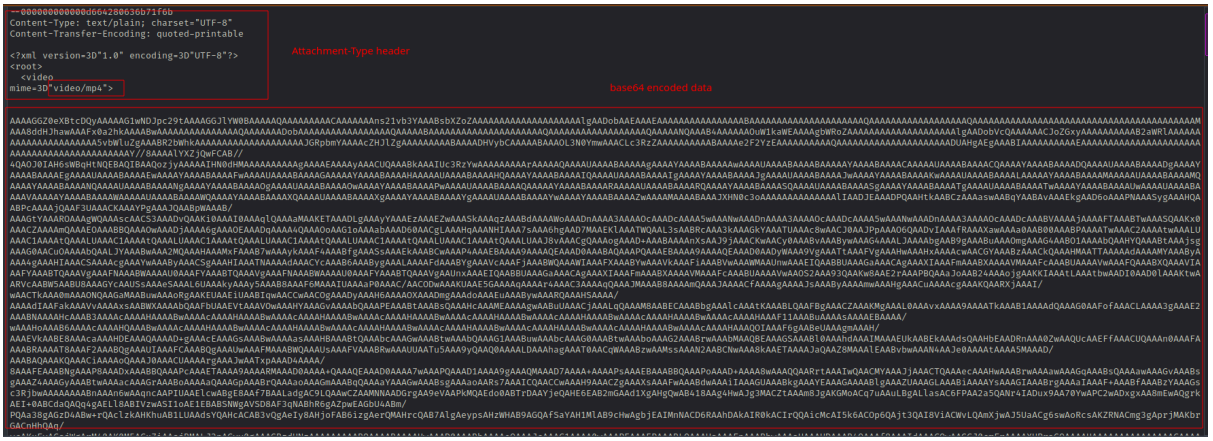


Figure 11: look_at_this_email.eml content

After decoding, the attachment was identified as a video file, which became the focus of further analysis. (Jody-admin, 2025)

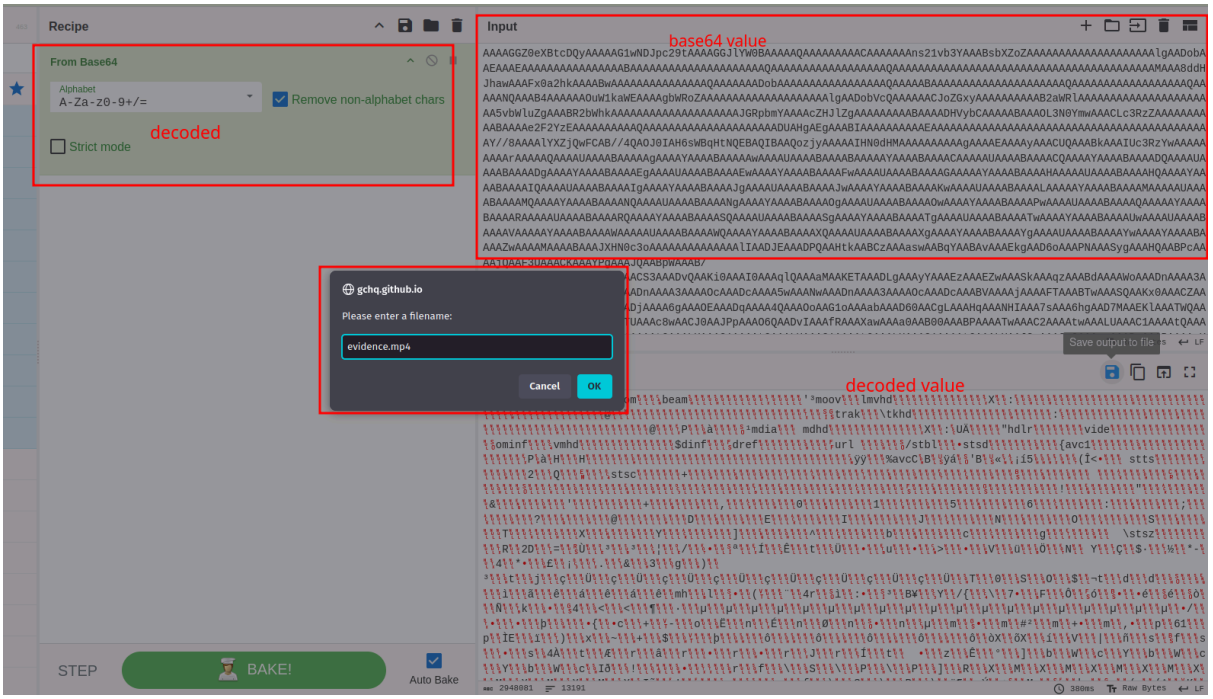


Figure 12: base64 to video

4.6. Extraction of the First Flag

The video file extracted from the second email was analyzed, and the required flag was successfully obtained at 13 seconds into the video. This process demonstrated the attacker's use of layered obfuscation and multiple communication channels to deliver key evidence and objectives in the breach scenario.

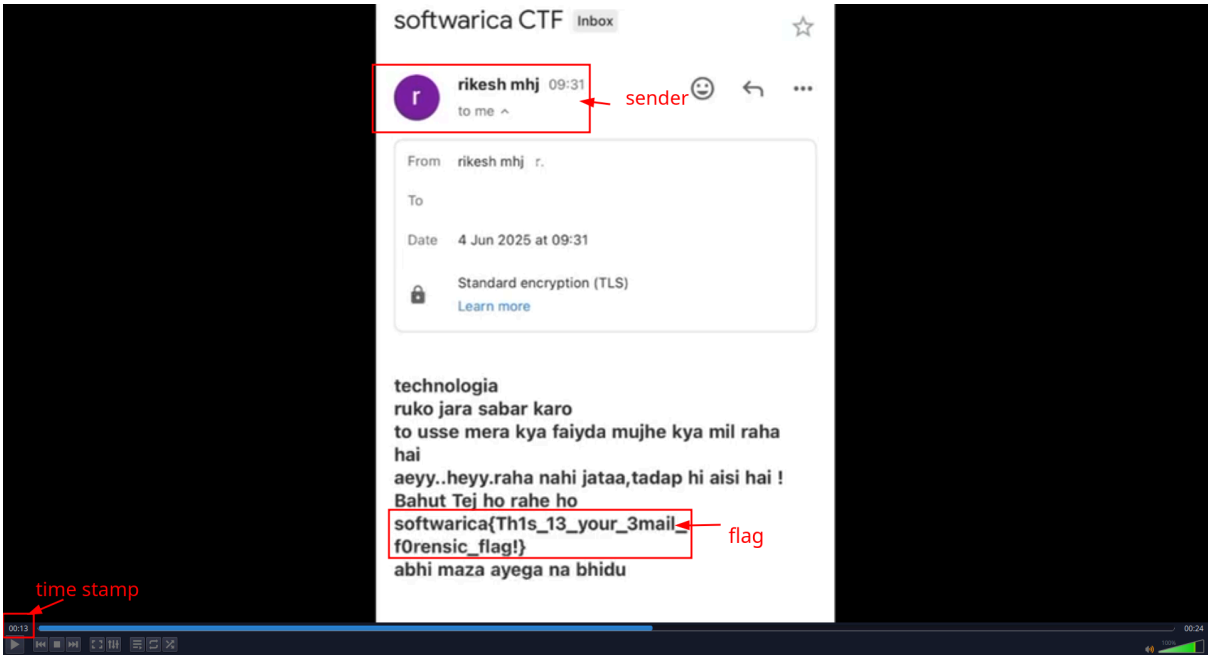


Figure 13: Flag

4.7. Extraction of Second Flag

After doing inspection of the email the message id looks suspicious, looking after it, it was found that the value is SHA-256 hash.

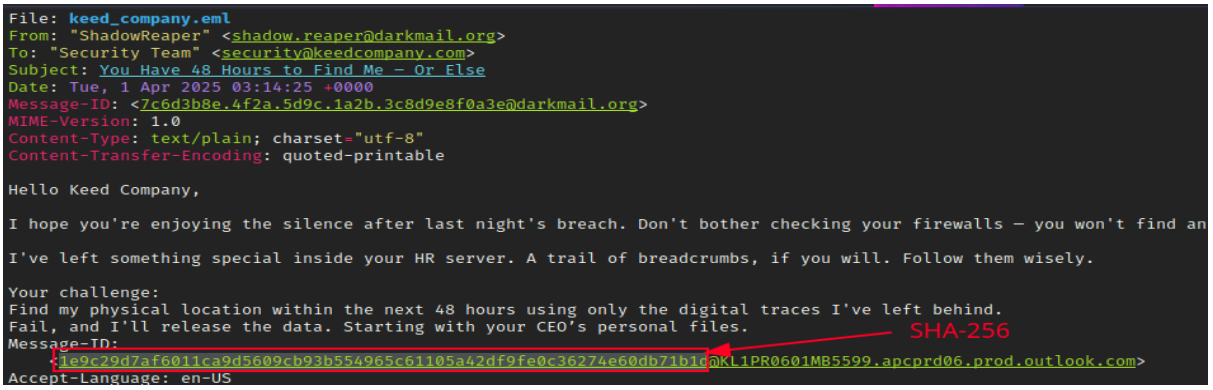


Figure 14: SHA-256 Hash

Then the hash value was copied and pasted into virustotal to carry further investigation, virustotal gave the result and that was a malicious file called xlm.txt. (Virustotal, 2024)

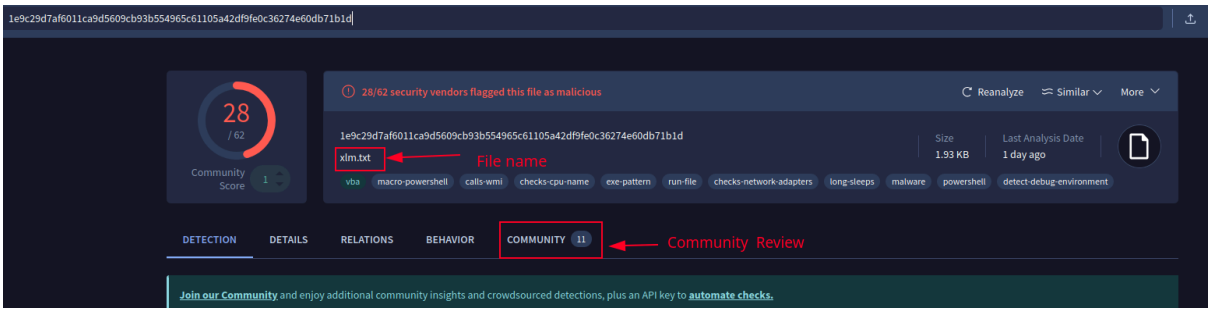


Figure 15: Malicious File

Seeing its community review, found the suspicious comment, the value was inserted at the end of the script.

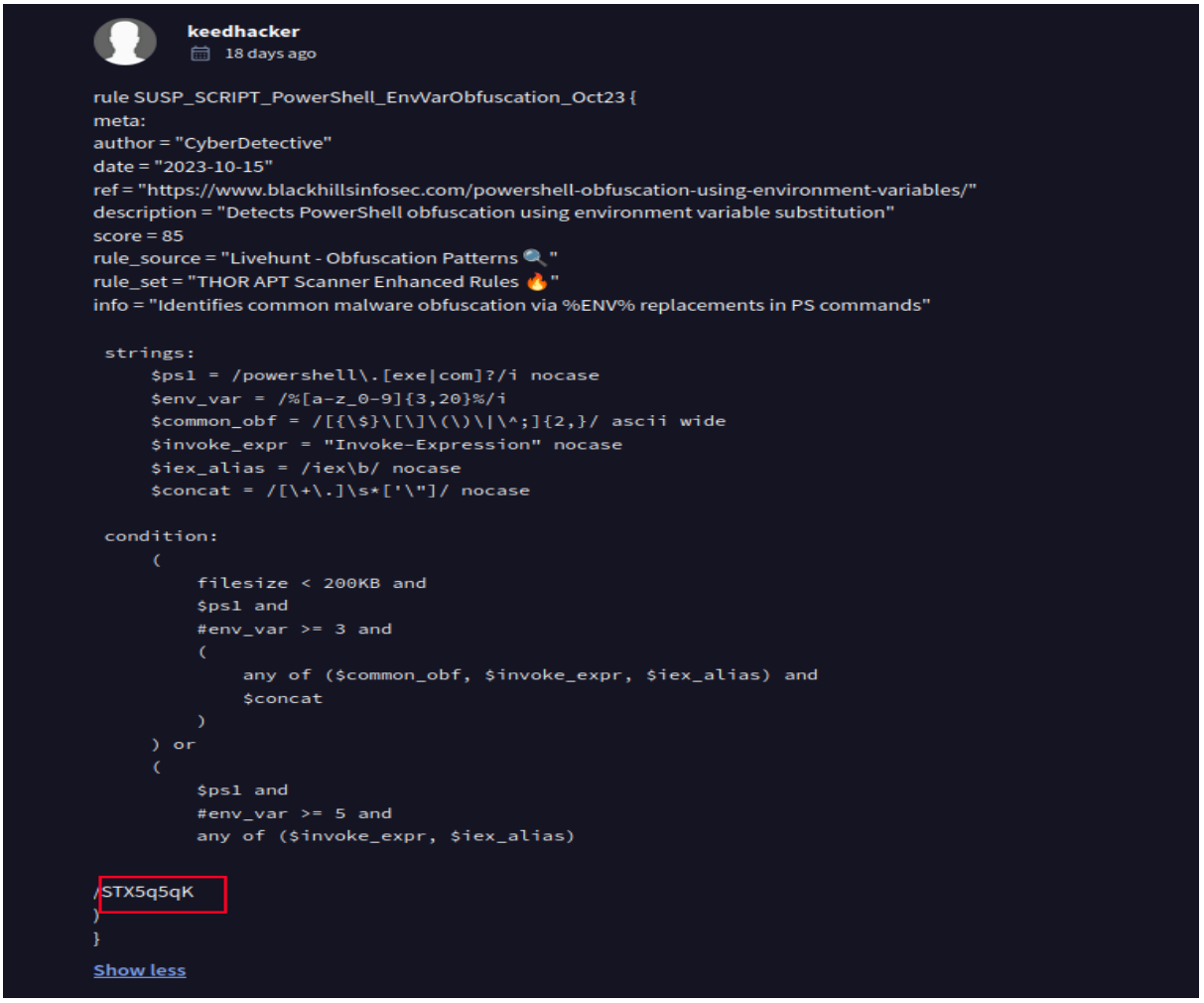


Figure 16: Suspicions data

It looks like some ciphered text, after copying that value in the cipher identifier, it gave a pastebin link. (Dcodefr, 2015)

Figure 17: Cipher identifier

The pastebin link was from KEEDHACKER. In that pastebin there was another link.

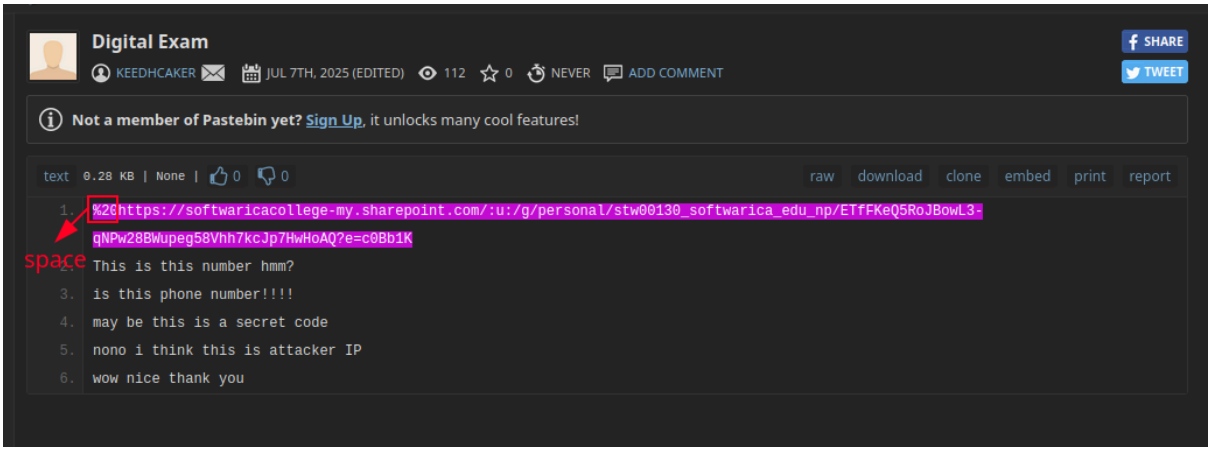


Figure 18: Pastebin link

It tooks to zip file which has an image.

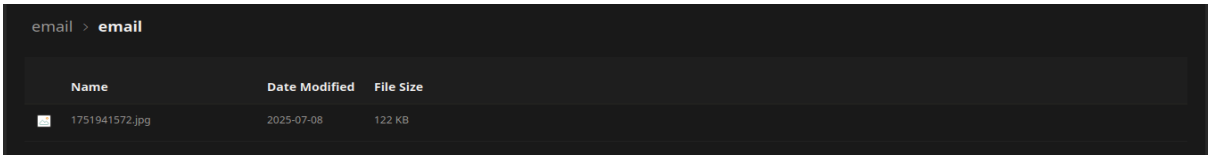


Figure 19: Zip file

After downloading that zip, it was found that it was password protected. So, to crack the password a tool john-the-ripper was used. (Openwall, 2017)

```
nott@honey in ~/Downloads via  v3.13.3 took 0s
λ zip2john email.zip > ha.txt
email.zip/email/ is not encrypted!
ver 1.0 email.zip/email/ is not encrypted, or stored with non-handled compression type
ver 2.0 efh 5455 efh 7875 email.zip/email/1751941572.jpg PKZIP Encr: 2b chk, TS_chk, cmplen=124823, decmplen=125035, crc=FB269D1C

nott@honey in ~/Downloads via  v3.13.3 took 0s
λ john --wordlist=/usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt ha.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
email            (email.zip/email/1751941572.jpg)
1g 0:00:00:00 DONE (2025-07-26 17:15) 20.00g/s 983040p/s 983040c/s 983040C/s 270489..triplet
Use the "--show" option to display all of the cracked passwords reliably
Session completed

nott@honey in ~/Downloads via  v3.13.3 took 2s
λ
```

Figure 20: Cracking zip password

After cracking the password, the image was inspected and to find the meta data of an image, a tool exiftool was used and it took to a a GPS coordinates. (Harvey, 2025)

```
nott@honey in ~/Downloads via  v3.13.3 took 2s
λ exiftool email/1751941572.jpg
ExifTool Version Number         : 13.30
File Name                       : 1751941572.jpg
Directory                       : email
File Size                       : 125 kB
File Modification Date/Time     : 2025:07:08 09:53:01+05:45
File Access Date/Time           : 2025:07:08 09:53:01+05:45
File Inode Change Date/Time     : 2025:07:26 17:15:29+05:45
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 1
Y Resolution                    : 1
Resolution Unit                 : None
Artist                          : 276062995 853293920   Coordinates
Y Cb Cr Positioning             : Centered
Image Width                     : 1024
Image Height                    : 1024
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 1024x1024
Megapixels                      : 1.0
```

Figure 21: GPS Coordinates

Afterward, it was a location of bajrabarahi temple, and after seeing the review there was an account found to be suspicious, the hex value and secret code was within the review.

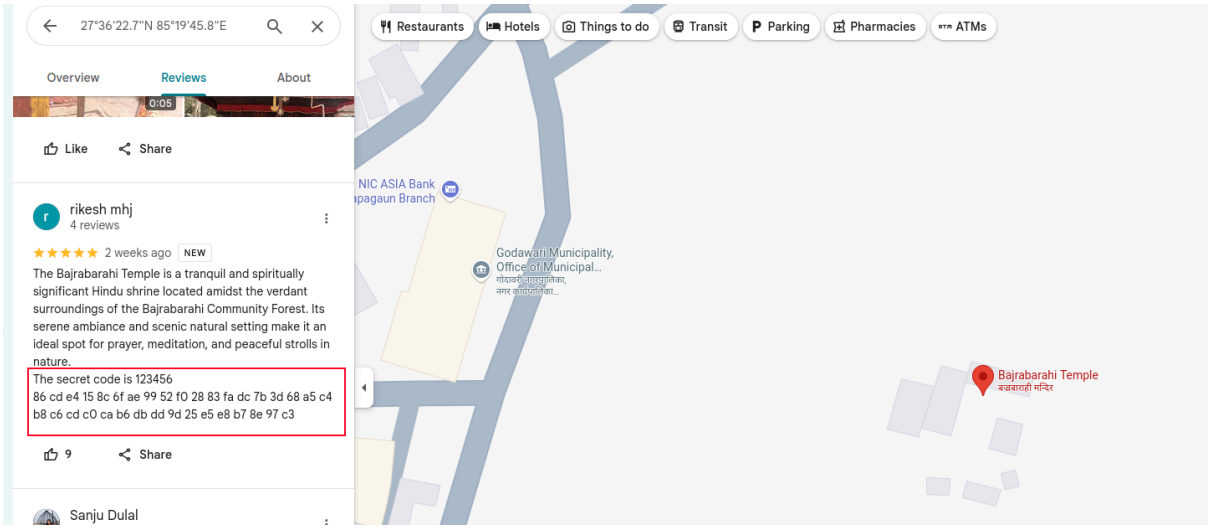


Figure 22: Cipher data

Again a tool cipher identifier was used to get the cipher data. And RC4 cipher was found.

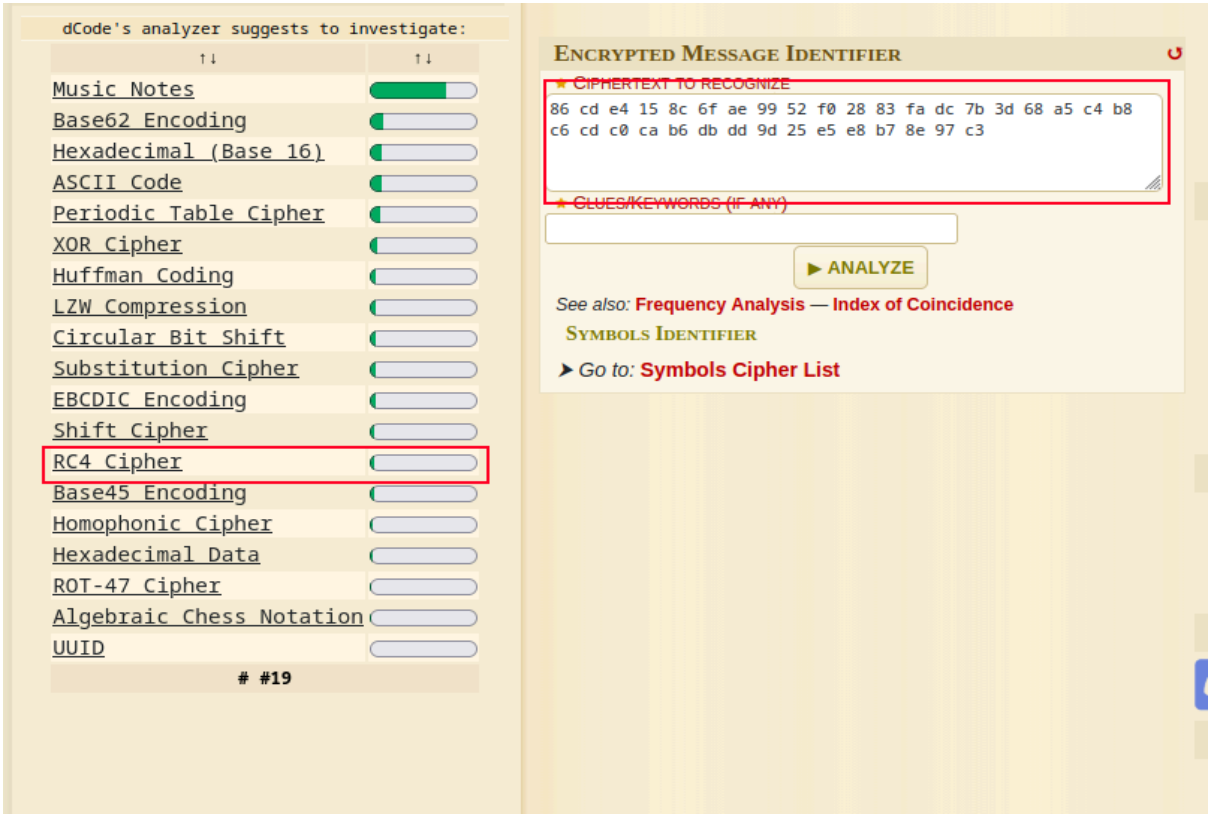


Figure 23: Found RC4 Cipher

After decoding that cipher from Cyberchef, the flag was extracted successfully. Flag was "**softwarica{3m41l_f0r3nsic_w4s_c00l}**"

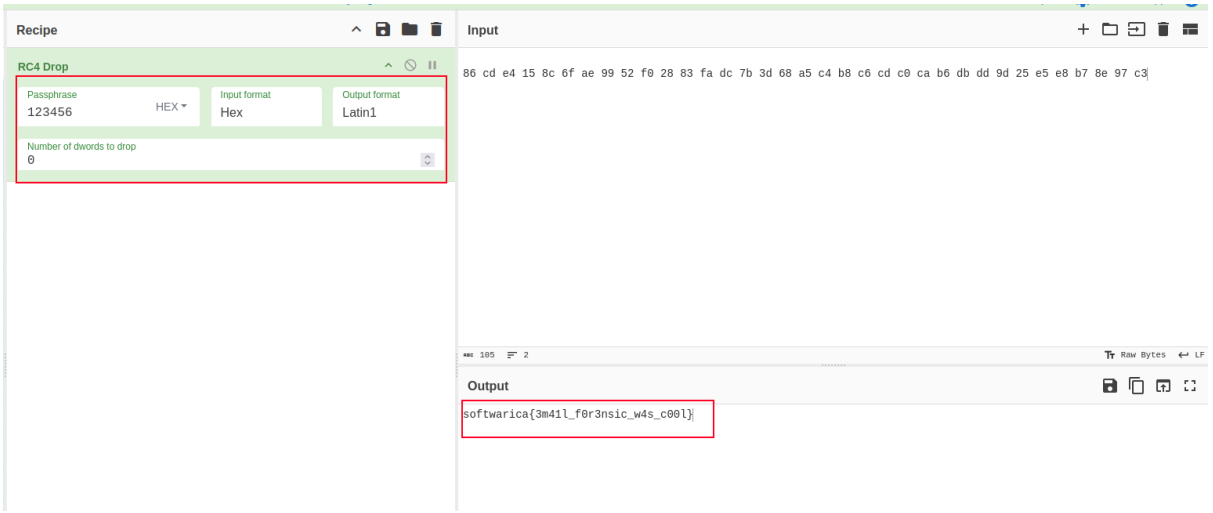


Figure 24: Second Flag of Email Forensics

4.8. Summary of Findings

The Email System Investigation determined that the breach began with a threatening email featuring a purposefully contaminated PDF attachment, which, after cleaning, disclosed a secret Google Drive link leading to a second email with a base64-encoded video file containing a flag. Further research identified a suspicious SHA-256 hash in the email header tied to a malware script with PowerShell obfuscation. This led to discovery of a password-protected ZIP file accessed via a pastebin link; after cracking the ZIP password, an image file's metadata gave GPS coordinates and buried hexadecimal cipher data. Decrypting the RC4-encrypted cipher revealed a second flag, indicating the attacker's sophisticated multi-stage obfuscation and covert communication strategies to prevent discovery and confound forensic operations.

5. Task 4: Network Traffic Analysis

5.1. Scope and Objectives

The investigation into Keed Loan Financial's data breach began with the discovery of a suspicious file named **keed_company.pcap** was found. The primary objective was to determine the attack vectors and attacker initial access to the server. (Wireshark, 2020)

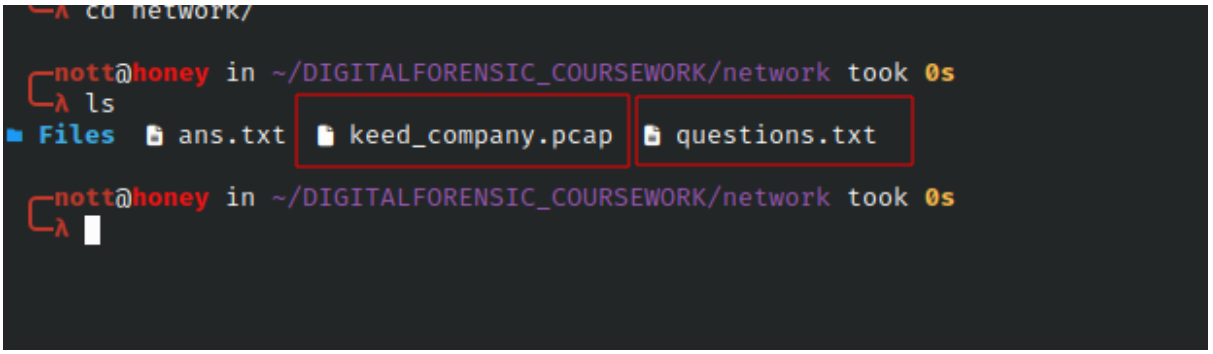


Figure 25: Pcap file

5.2. FTP

The initial attack was from the ftp, where ftp was misconfigured with **anonymous** users, so that anyone can have access to the ftp. So, here attacker found this and get into the ftp server.

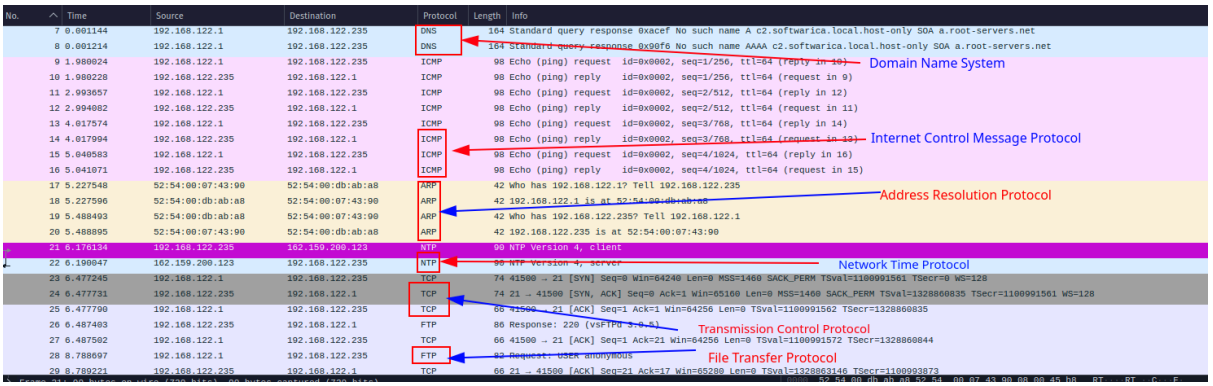


Figure 26: Protocols

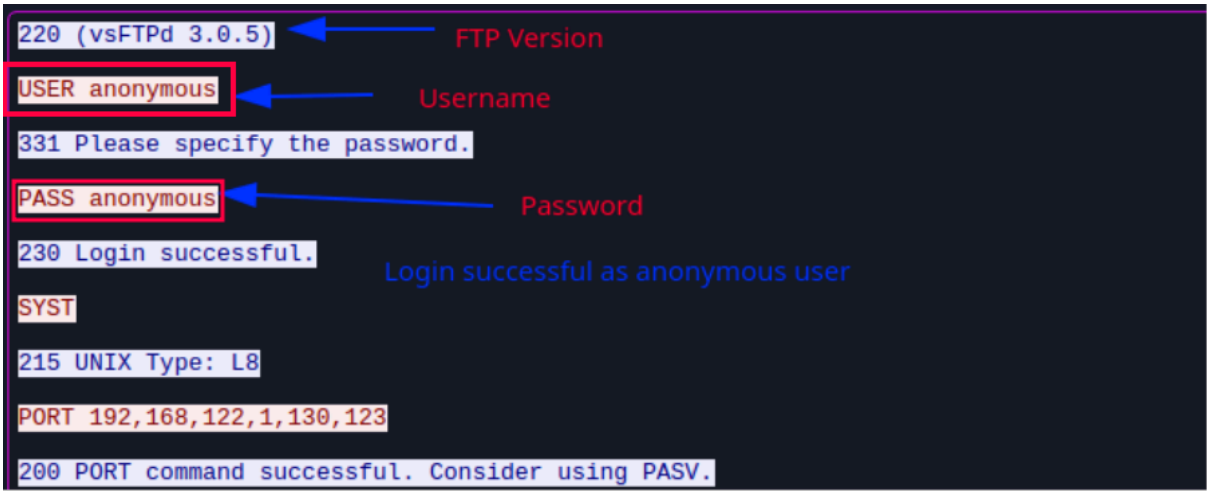


Figure 27 : Ftp Login

From here attacker download these **files flag.txt, meme.png, one of this is flag.pdf, plane.mp3** and **share.zip**

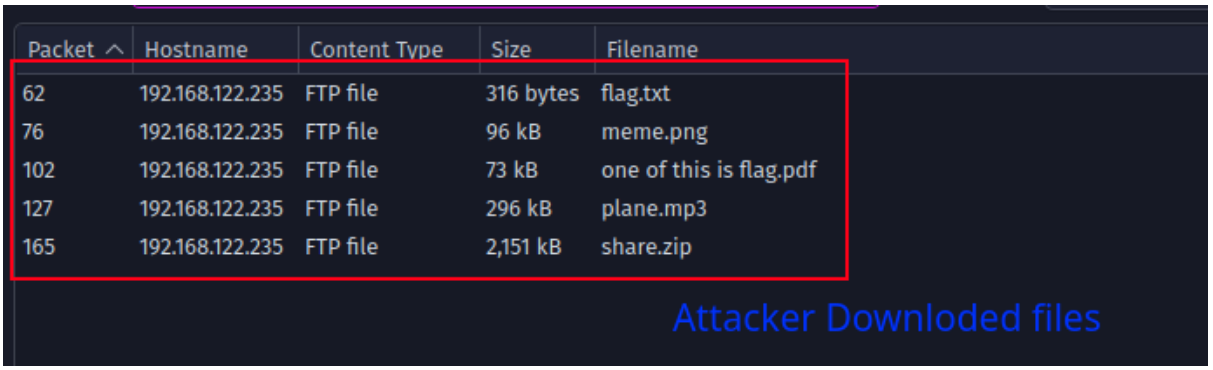


Figure 28: Downloaded files

After downloading these into my local machine I found the flag format.

Figure 29: Flag format

After using a lot of steganography tools to extract hidden data in meme.png, nothing I got from meme.png and at last I used binwalk to extract the embedded file. (Linuxcommandlibrary, 2020)

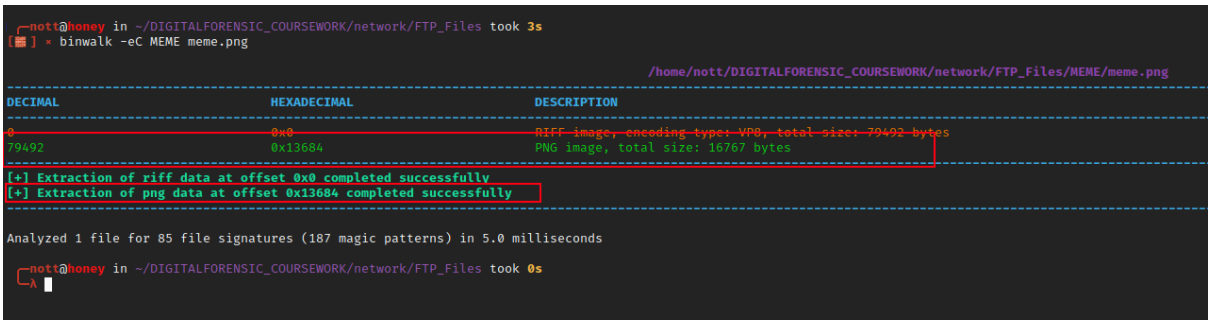


Figure 30: Extracted embedded file

After viewing that extracted file got the next flag

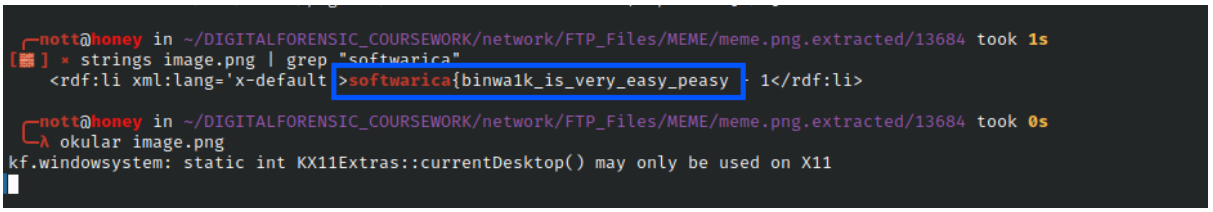


Figure 31: Image flag

There was a pdf whose name was *'one of this is flag.pdf'*. So after viewing that pdf there was nothing except of fake flag but in last there was something written in newari after translating that sentences it was, try another tools or use net was told. So, after using different tools, obtained that flag.

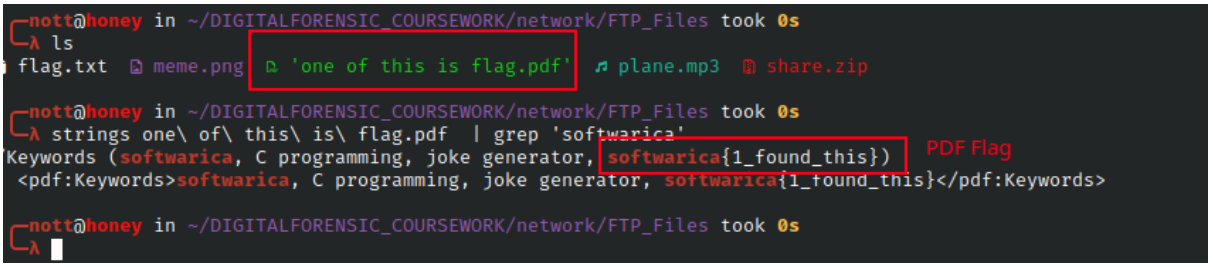


Figure 32: Pdf flag

The plane.mp3 was an audio file so, what's in there makes us curious after hearing that audio, found the next flag at 26 sec.

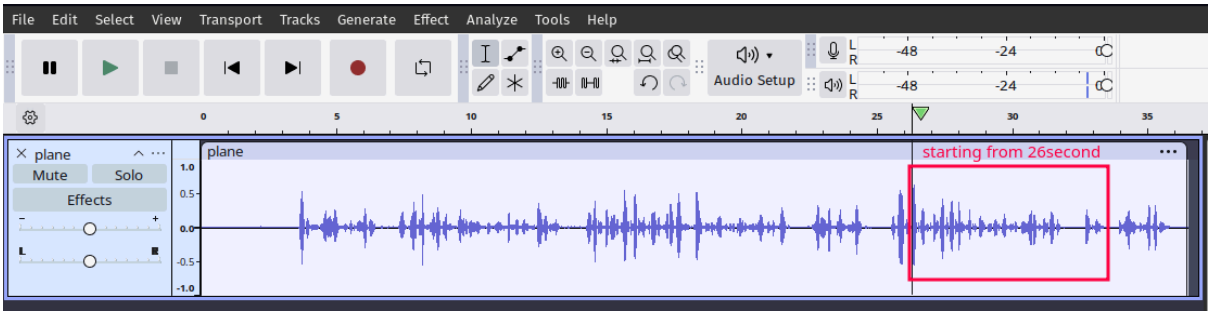


Figure 33: Plane.mp3 Flag

And there was a file which was zip that is share.zip, while doing unzip it was skipping files, then use 7z which support modern zipped file, but it was asking password then used zip2john to create a has and then using john bruteforce helped to find the password.

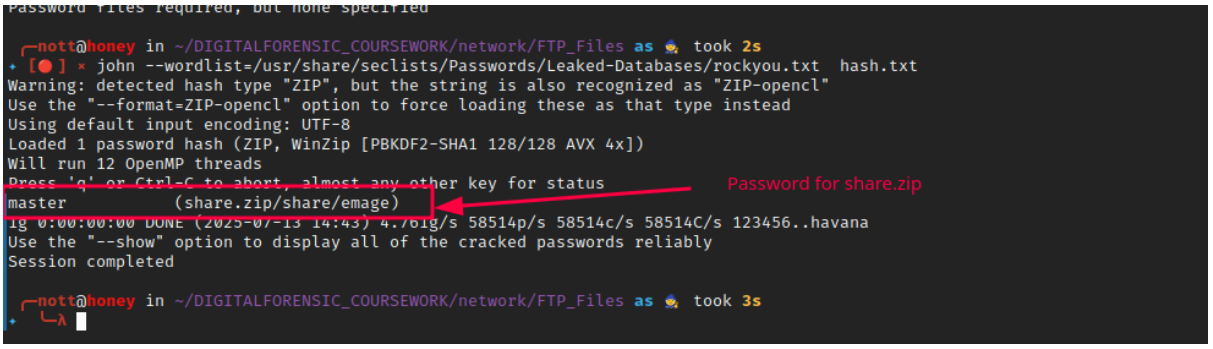


Figure 34: hash crack

It consists these files, **emage, email1.em1, game.exe and part2.pcap**

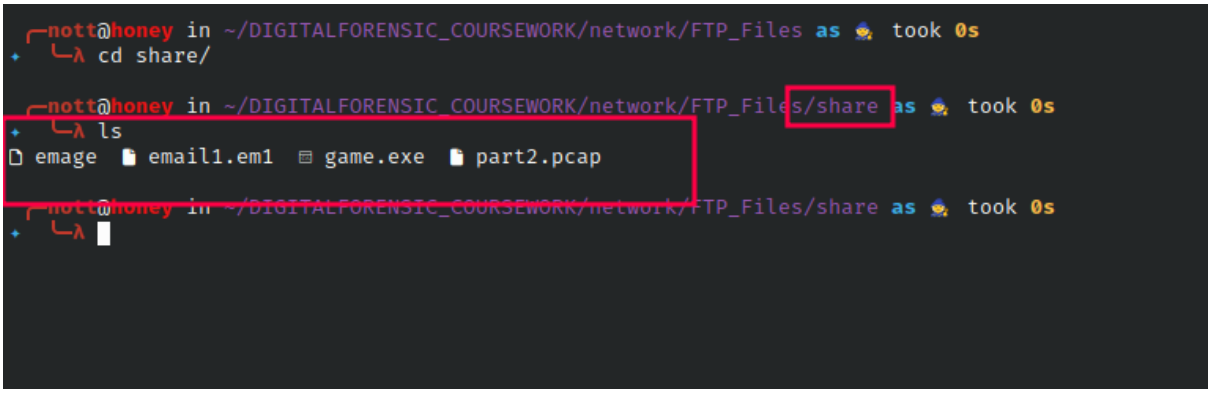


Figure 35: Unzipped share.zip

After viewing it one by one the there was a file called email1.em1 consist secret value into base64 encoded after decoding it gives a flag.

```
┌─nott@honey in ~/DIGITALFORENSIC_COURSEWORK/network/FTP_Files/share as 🧙 took 0s
└─λ echo  'c29mdHdhcmljYXtlbWFpbF9mb3JlbnNpY18xc19mdW59' | base64 -d
softwarica{email_forensic_1s_fun}   email flag
┌─nott@honey in ~/DIGITALFORENSIC_COURSEWORK/network/FTP_Files/share as 🧙 took 0s
└─λ ls
```

Figure 36: email1.em1 flag

And the next interesting thing was game.exe, after viewing with this from strings command, manifest script for windows, then grabbing its hash confirms it's a malware which tagname was "**amadey**".

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity
    type="win32"
    name="StartIsBack installation / configuration utility"
    version="1.0.0.0"
    processorArchitecture="*"/>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity
        type="win32"
        name="Microsoft.Windows.Common-Controls"
        version="6.0.0.0"
        publicKeyToken="6595b64144ccf1df"
        language="*"
        processorArchitecture="*"/>
    </dependentAssembly>
  </dependency>
</assembly><?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level='asInvoker' uiAccess='false' />
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Figure 37: game.exe

```
┌─nott@honey in ~/DIGITALFORENSIC_COURSEWORK/network/FTP_Files/share took 0s
└─λ md5sum game.exe
aa883f75bff0257a0fefd5d8d20c6297  game.exe
```

Figure 38: md5sum of game.exe

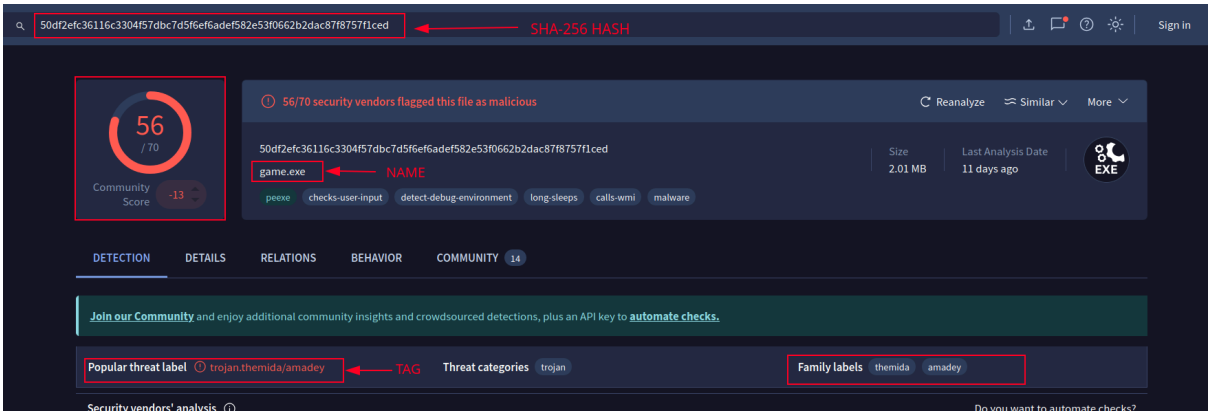


Figure 39: Popular tag

5.3. HTTP

In the second pcap file called **"part2pcap"** all the **http** traffic was inspected. The initial foothold was the attacker login using **"username:tryhackme&passwd:!@#$%^&*".**

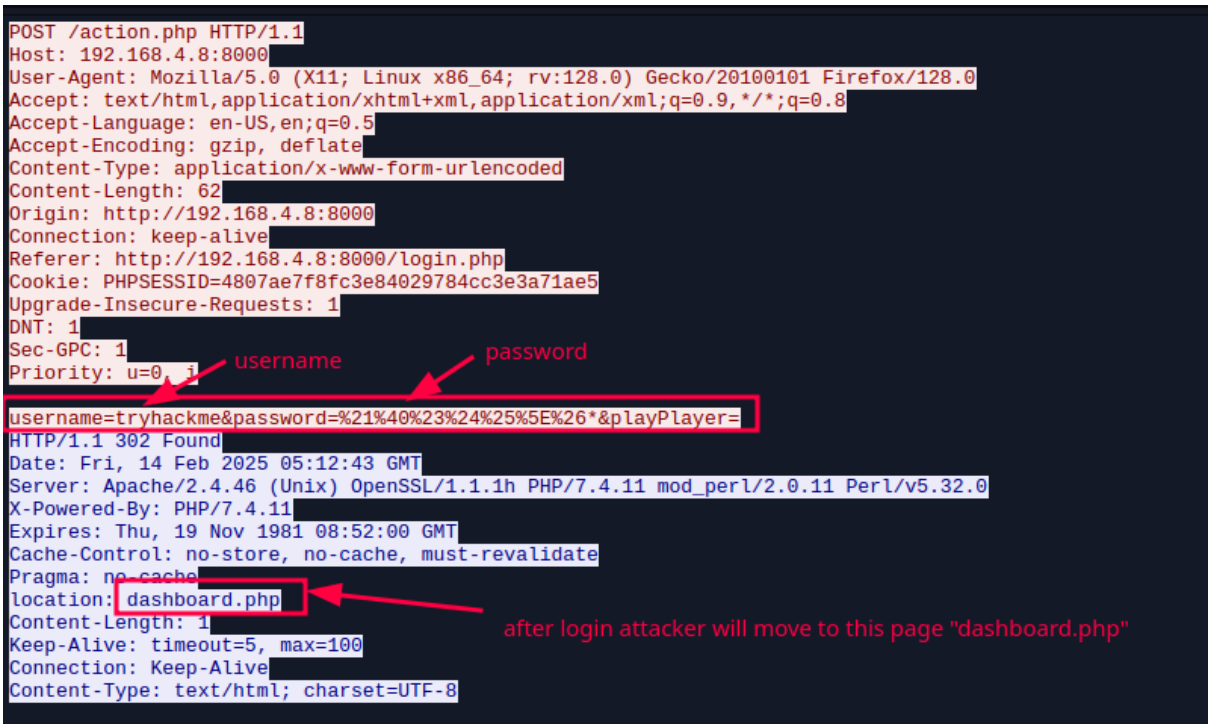


Figure 40: Login attempt

After more digging up, attacker tries a lot of payload to gain access to the server, but the vulnerability that the attacker found was **LIF(Local File Inclusion)** and the payload is **"../../../../../../../etc/passwd"** and the attacker get the user accounts information.

Figure 41: LFI Vulnerability

While the attacker found LFI, the server has more vulnerability so, the threat actor found file upload bug and the flag is **"softwarica{file_upl0ad_bug}".** Here the attacker tries to upload a web shell but it does not work.

Figure 42:File upload Flag

After knowing there was an upload vulnerability, the threat actor tried to upload a reverse shell into the server and the payload worked and got the reverse shell through "**../challenges/1739510993rev.phtml"** this payload. Excessing that uploaded reverse shell file from **LFI.**

Figure 43: Final Payload

After analysing the tcp stream to that upload endpoint where attacker upload reverse file there was a flag **"softwarica{reverse_f1ag}".**

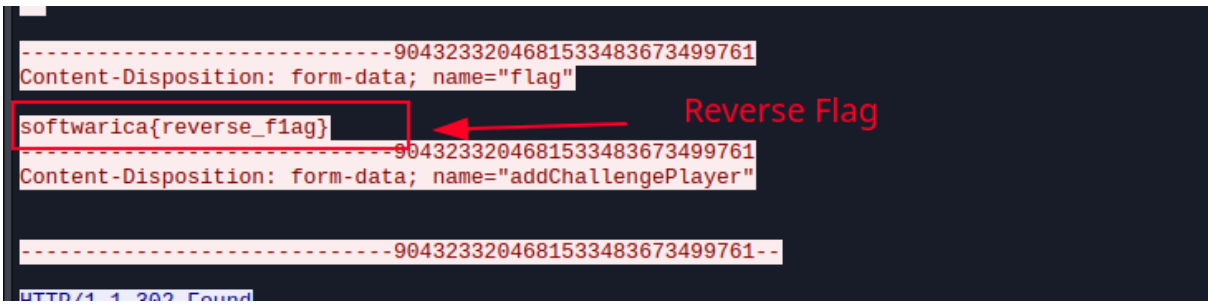


Figure 44: Reverse Flag

Gaining access to the shell, the root flag was base64 encoded. After decoding that encoded data, successfully retrieve the flag.

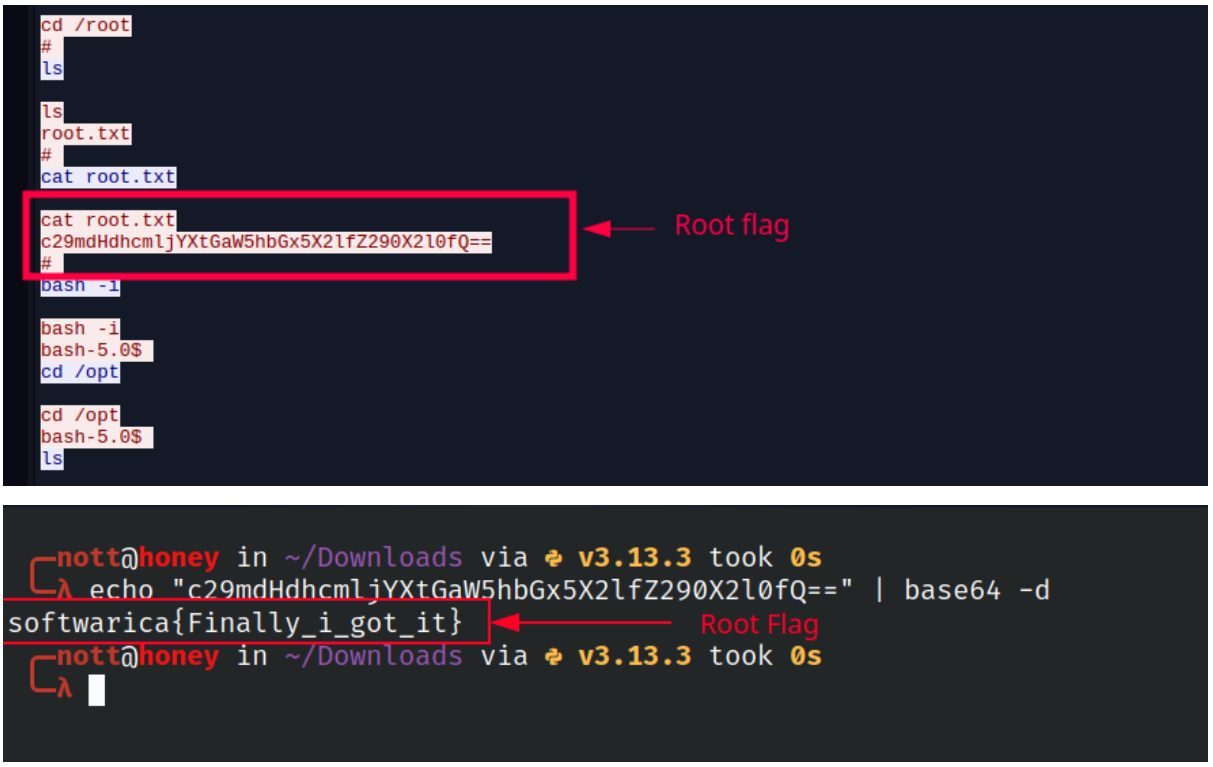


Figure 45: Root Flag

After gaining shell access, the attacker downloaded a file named **"installer.elf"** from an HTTP server located at "**http://192.168.4.5:7777/installer.elf".** For forensic analysis, the same file was obtained by the investigation team, and its hash value was calculated to verify its integrity.

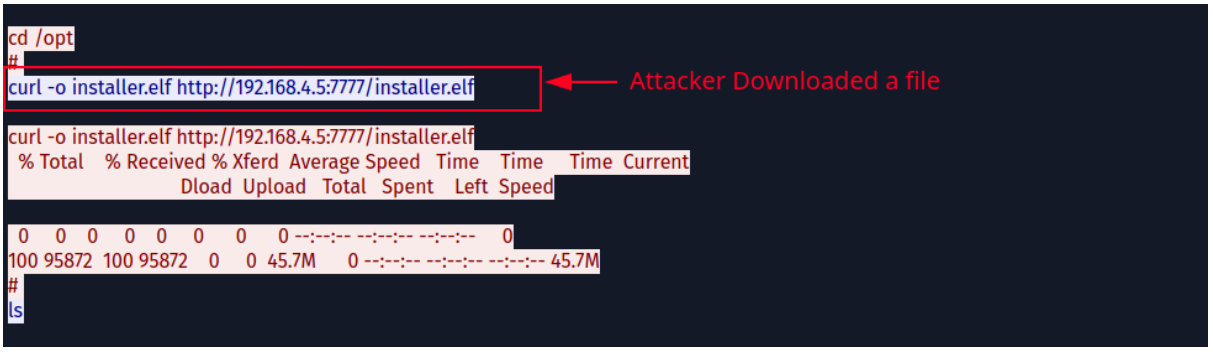


Figure 46: Installed file

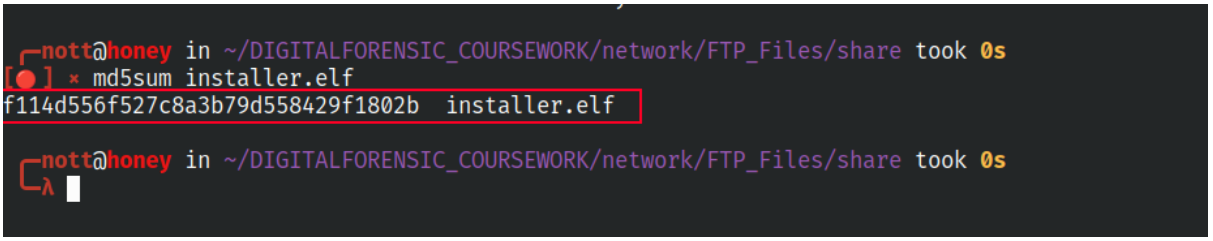


Figure 47: MD5 HASH of installer.elf

After calculating the hash of the downloaded installer.elf file, the forensic team used VirusTotal to verify whether the file was malicious. The scan results identified the file as malware, tagged under the **Mirai** botnet family.

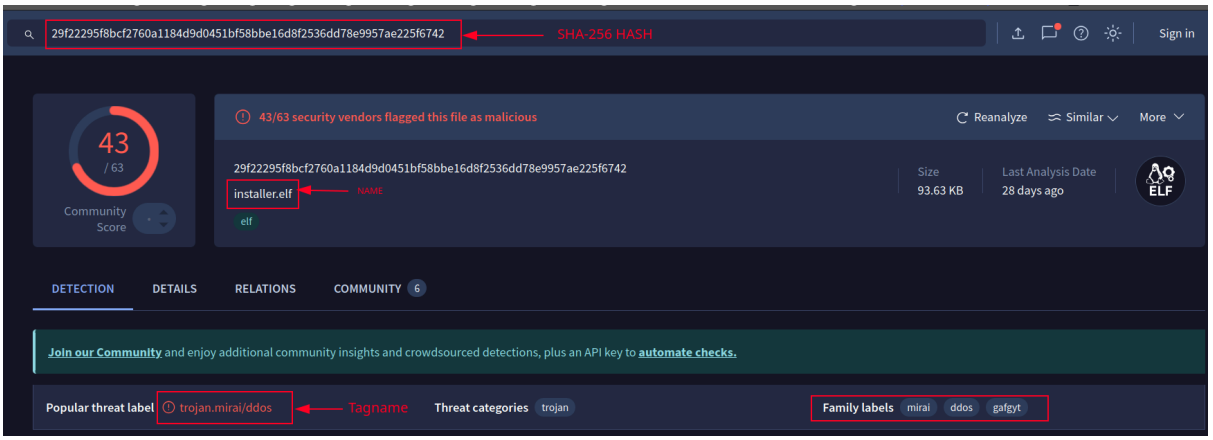


Figure 48: Tagname Mirai

Attacker also found sql injection on **/search.php?query**

Format **/search.php:query**

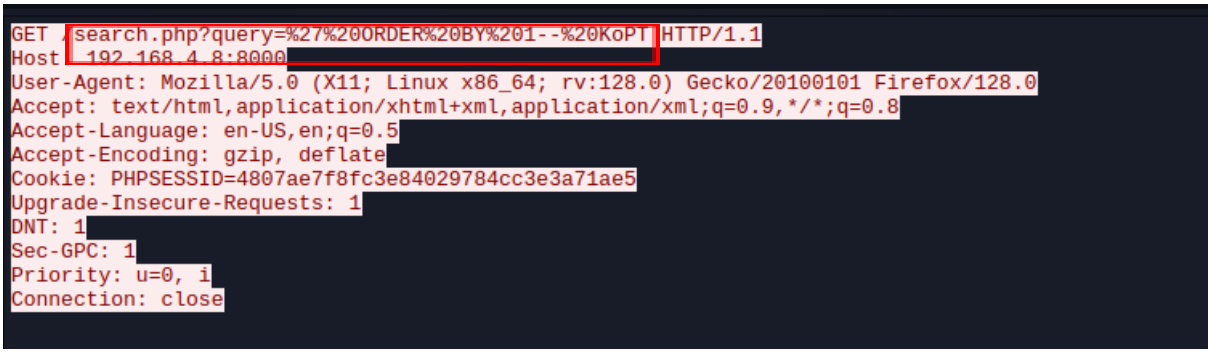


Figure 49: SQL Attack

5.4. Summary of Findings

The keed_company.pcap and part2.pcap files from the network forensic analysis showed that Keed Loan Financial was the victim of a multi-phase cyberattack. Using steganography and encoding, the attacker originally acquired access to an FTP server that was misconfigured to support anonymous login, which allowed them to download files carrying disguised flags. SQL injection, file upload failures, and Local File Inclusion (LFI) are among the HTTP-based web vulnerabilities that the attacker later assailed. To find these exploits, extensive packet analysis of HTTP traffic was employed. After a successful reverse shell upload using LFI allowed remote access to the server, a malicious application (installer.elf) was downloaded; VirusTotal analysis and hash verification subsequently revealed that the virus was part of the Mirai botnet. The network data also showed brute-force login attempts and payload executions, revealing how unpatched vulnerabilities and inappropriate server setups were exploited to expand access and disseminate malware.

6. Task 5: Memory Forensics Analysis

6.1. Scope and Objectives

Forensic investigation of a volatile memory image acquired from a machine that may have been compromised is part of this process. Finding possible in-memory signs of malicious activity, such as questionable processes, executable payloads, and obfuscated data, is the main goal of the investigation. Recovery of hidden information is the major emphasis of the study, and special stress is paid to locating and deleting any embedded flags or artifacts that are preserved in memory-resident code or executables.

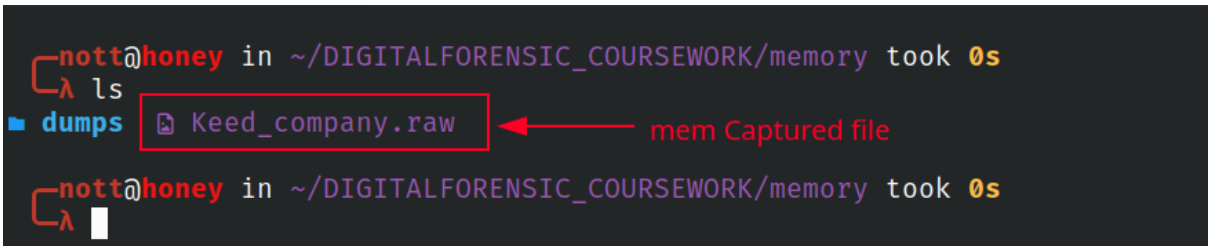


Figure 50: Memory Captured File

6.2. Extracting Flag

Initially, the strings command was applied to search the acquired memory file's data for any ASCII sequences or patterns that might be read. During this technique, a number of strings that resembled the phrase "softwarica" were noticed. These patterns were further investigated and filtered using the grep software. These strategies helped uncover a pattern that looked like a flag format. The exact flag embedded in the memory capture was established after some examination.

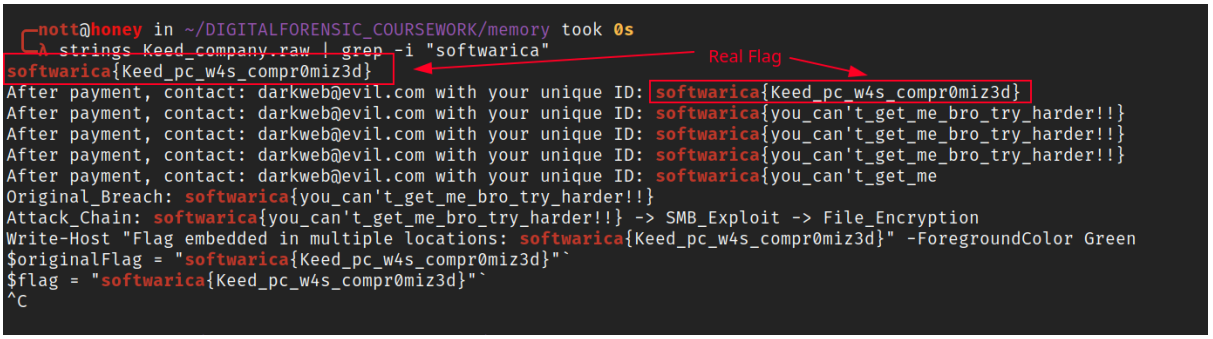


Figure 51: Memory Flag-1

After performing this and that, memory forensics was done using a tool called volatility. The imageinfo plugin first analyzes a raw memory image to identify the most likely operating system and version at the time the image was taken. Additionally, a pre-defined set of configuration and symbols known as the profile is used to map particular OS versions, architectures (x86/x64), and Service Pack levels. (Jekyll, 2024)

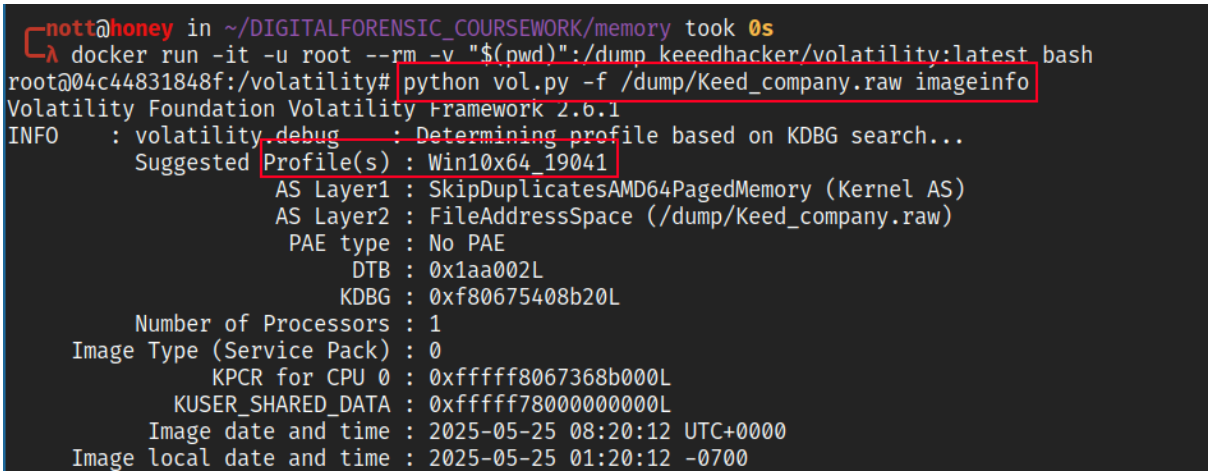


Figure 52: Finding imageinfo

A tree-structured listing of the processes simplifies the identification of child processes and their trail back to their parent, which helps identify the source of suspicious activity. When the process tree was studied, multiple child processes that were weird or unexpected were uncovered; this makes it possible that a parent process launched the possibly risky conduct.

```
root@04c44831848f:/volatility# python vol.py -f /dump/Keed_company.raw --profile=Win10x64_19041 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xffff9e0ec2947140:services.exe                      604    512      8      0 2025-05-25 20:57:13 UTC+0000
. 0xffff9e0ec30472c0:svchost.exe                      836    604     15      0 2025-05-25 20:57:13 UTC+0000
. 0xffff9e0ec43680c0:svchost.exe                     3332    604      4      0 2025-05-25 08:17:08 UTC+0000
. 0xffff9e0ec31712c0:svchost.exe                     1064    604     24      0 2025-05-25 20:57:14 UTC+0000
. 0xffff9e0ec2008240:svchost.exe                       52    604     66      0 2025-05-25 20:57:14 UTC+0000
.. 0xffff9e0ec16af080:MicrosoftEdgeU                 2216     52      4      0 2025-05-25 08:16:46 UTC+0000
.. 0xffff9e0ec31d9080:sihost.exe                      256     52     14      0 2025-05-25 08:16:45 UTC+0000
... 0xffff9e0ec1df9340:olk.exe                       4116    256     28      0 2025-05-25 08:19:02 UTC+0000
.... 0xffff9e0ec449b340:msedgewebview2               6568   4116     49      0 2025-05-25 08:19:03 UTC+0000
..... 0xffff9e0ec45350c0:msedgewebview2              6240   6568     19      0 2025-05-25 08:19:04 UTC+0000
..... 0xffff9e0ec498a0c0:msedgewebview2              6256   6568      8      0 2025-05-25 08:19:04 UTC+0000
..... 0xffff9e0ec47c90c0:msedgewebview2              5864   6568     18      0 2025-05-25 08:19:04 UTC+0000
..... 0xffff9e0ec3d1a080:msedgewebview2              3228   6568     17      0 2025-05-25 08:19:09 UTC+0000
..... 0xffff9e0ec16ae300:msedgewebview2              6628   6568      8      0 2025-05-25 08:19:03 UTC+0000
.. 0xffff9e0ec155a0c0:taskhostw.exe                   436     52      9      0 2025-05-25 08:16:45 UTC+0000
... 0xffff9e0ec2942080:wininit.exe                    512    436      1      0 2025-05-25 20:57:12 UTC+0000
.... 0xffff9e0ec2971080:lsass.exe                     612    512     10      0 2025-05-25 20:57:13 UTC+0000
.... 0xffff9e0ec2164080:fontdrvhost.ex                748    512      5      0 2025-05-25 20:57:13 UTC+0000
... 0xffff9e0ec28530c0:csrss.exe                      444    436      9      0 2025-05-25 20:57:12 UTC+0000
.. 0xffff9e0ec31d5300:consent.exe                    6468     52      8      0 2025-05-25 08:19:22 UTC+0000
. 0xffff9e0ec3279300:svchost.exe                     1616    604     11      0 2025-05-25 08:12:17 UTC+0000
.. 0xffff9e0ec3cf5080:audiodg.exe                    4796   1616      5      0 2025-05-25 08:18:53 UTC+0000
. 0xffff9e0ec3624340:svchost.exe                     2164    604     13      0 2025-05-25 08:12:19 UTC+0000
. 0xffff9e0ec31d02c0:svchost.exe                     1180    604     20      0 2025-05-25 20:57:14 UTC+0000
. 0xffff9e0ec1fa3300:svchost.exe                     3244    604     13      0 2025-05-25 08:16:49 UTC+0000
. 0xffff9e0ec3384240:svchost.exe                     1712    604      7      0 2025-05-25 08:12:18 UTC+0000
. 0xffff9e0ec339a2c0:svchost.exe                     1736    604      4      0 2025-05-25 08:12:18 UTC+0000
. 0xffff9e0ebcc76280:svchost.exe                     1232    604      3      0 2025-05-25 08:14:20 UTC+0000
. 0xffff9e0ec2195240:svchost.exe                      732    604     17      0 2025-05-25 20:57:13 UTC+0000
.. 0xffff9e0ec1975080:RuntimeBroker.                 3648    732      4      0 2025-05-25 08:16:52 UTC+0000
.. 0xffff9e0ec307c080:RuntimeBroker.                 4556    732     15      0 2025-05-25 08:16:59 UTC+0000
... 0xffff9e0ec1fd2080:powershell.exe                4428   4556     20      0 2025-05-25 08:19:27 UTC+0000
```

Figure 53: Listing Process

The abnormal processes are extracted and listed accordingly.

```
root@04c44831848f:/volatility# python vol.py -f /dump/Keed_company.raw --profile=Win10x64_19041 pstree | grep 'flag.exe\|ransomware.exe\|malware.exe\|save.exe\|DumpIt.exe'
Volatility Foundation Volatility Framework 2.6.1
WARNING : volatility.debug    : PID 612 PPID 512 has already been seen
WARNING : volatility.debug    : PID 748 PPID 512 has already been seen
... 0xffff9e0ec3bf82c0:ransomware.exe               7116   3032      1      0 2025-05-25 08:17:55 UTC+0000
... 0xffff9e0ec4122080:DumpIt.exe                   4312   3032      2      0 2025-05-25 08:20:10 UTC+0000
... 0xffff9e0ec1a6b240:save.exe                     2312   3032      1      0 2025-05-25 08:17:58 UTC+0000
... 0xffff9e0ebc6f9080:flag.exe                     6960   3032      1      0 2025-05-25 08:17:50 UTC+0000
... 0xffff9e0ec43d92c0:malware.exe                  7068   3032      1      0 2025-05-25 08:17:53 UTC+0000
root@04c44831848f:/volatility#
```

Suspicious Process

Figure 54: Suspicious process

The abnormal process was dumped to the local machine to further inspection.

```
root@04c44831848f:/volatility# python vol.py -f /dump/Keed_company.raw --profile=Win10x64_19041 procdump -p 6960,7116,4312,2312,7068  -D /dump/SUS_dump
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xffff9e0ebc6f9080 0x00007ff6f8d30000 flag.exe             OK: executable.6960.exe
0xffff9e0ec43d92c0 0x00007ff625590000 malware.exe          OK: executable.7068.exe
0xffff9e0ec3bf82c0 0x00007ff67dbf0000 ransomware.exe       OK: executable.7116.exe
0xffff9e0ec1a6b240 0x00007ff627ce0000 save.exe             OK: executable.2312.exe
0xffff9e0ec4122080 0x00000000001b0000 DumpIt.exe           OK: executable.4312.exe
root@04c44831848f:/volatility#
```

PID

Specifying Directory

Process dumped

Figure 55: Dumping Process

In order to provide a higher-level, more understandable description of the program's behavior, the assembly code was decompiled using the Ghidra tool. After loading all suspicious

processes into Ghidra for thorough static analysis, the forensic team was able to look at the code structure, discover malicious logic, and hunt down any attack functions or embedded payloads. (Fox, 2023)

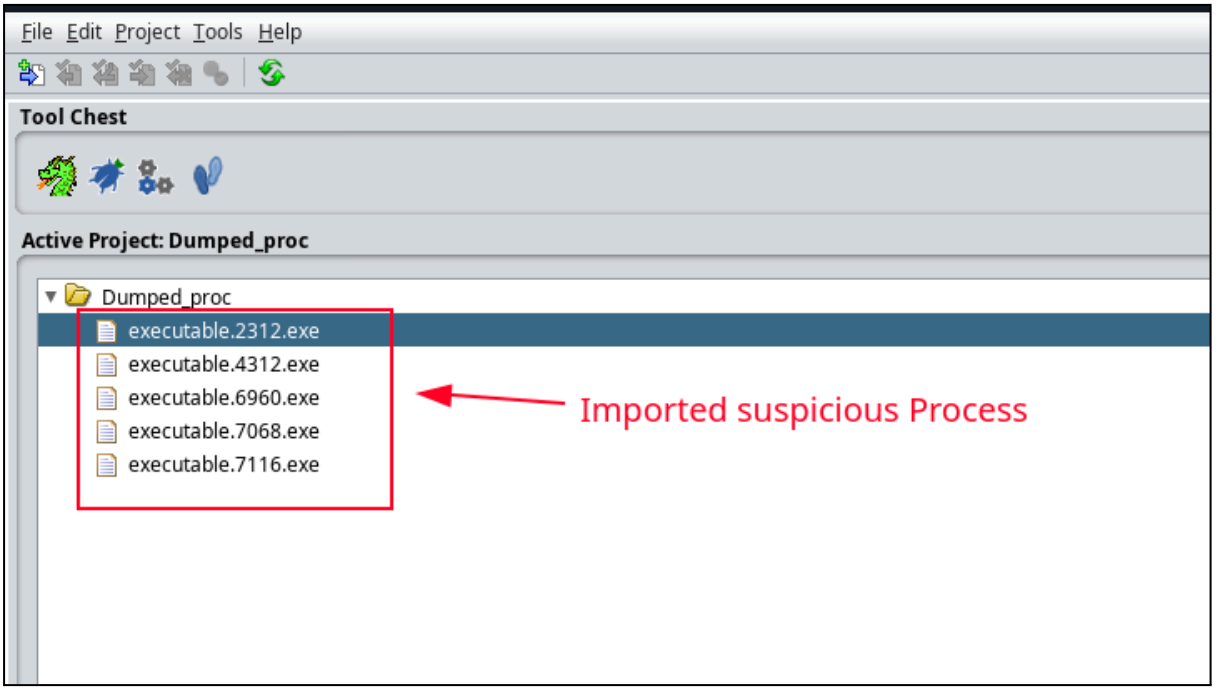


Figure 56: Imported Process

Following an inspection of every process detected, some of them indicated that the malicious file might include embedded flags. An in-depth investigation of the associated files revealed that the genuine flag was contained in the save.exe file (also known as executable.2312.exe). According to its behavior and content during forensic testing, this file was verified as the source.

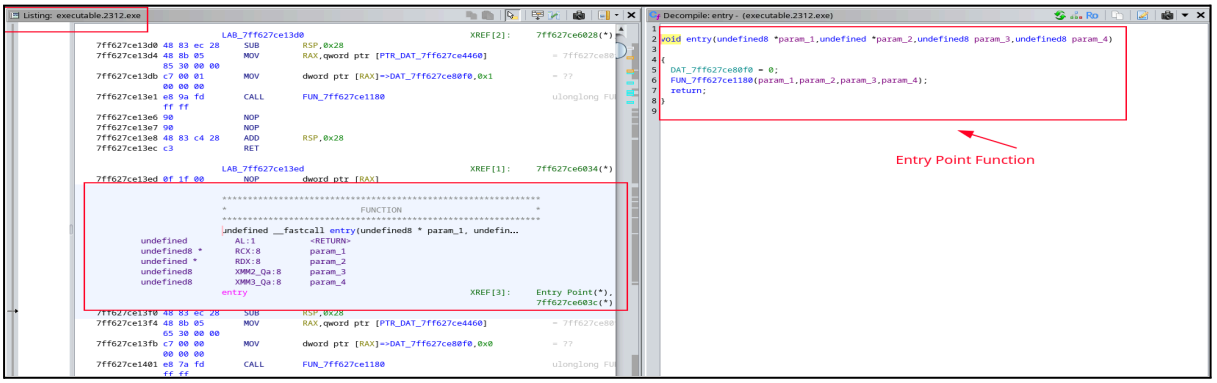
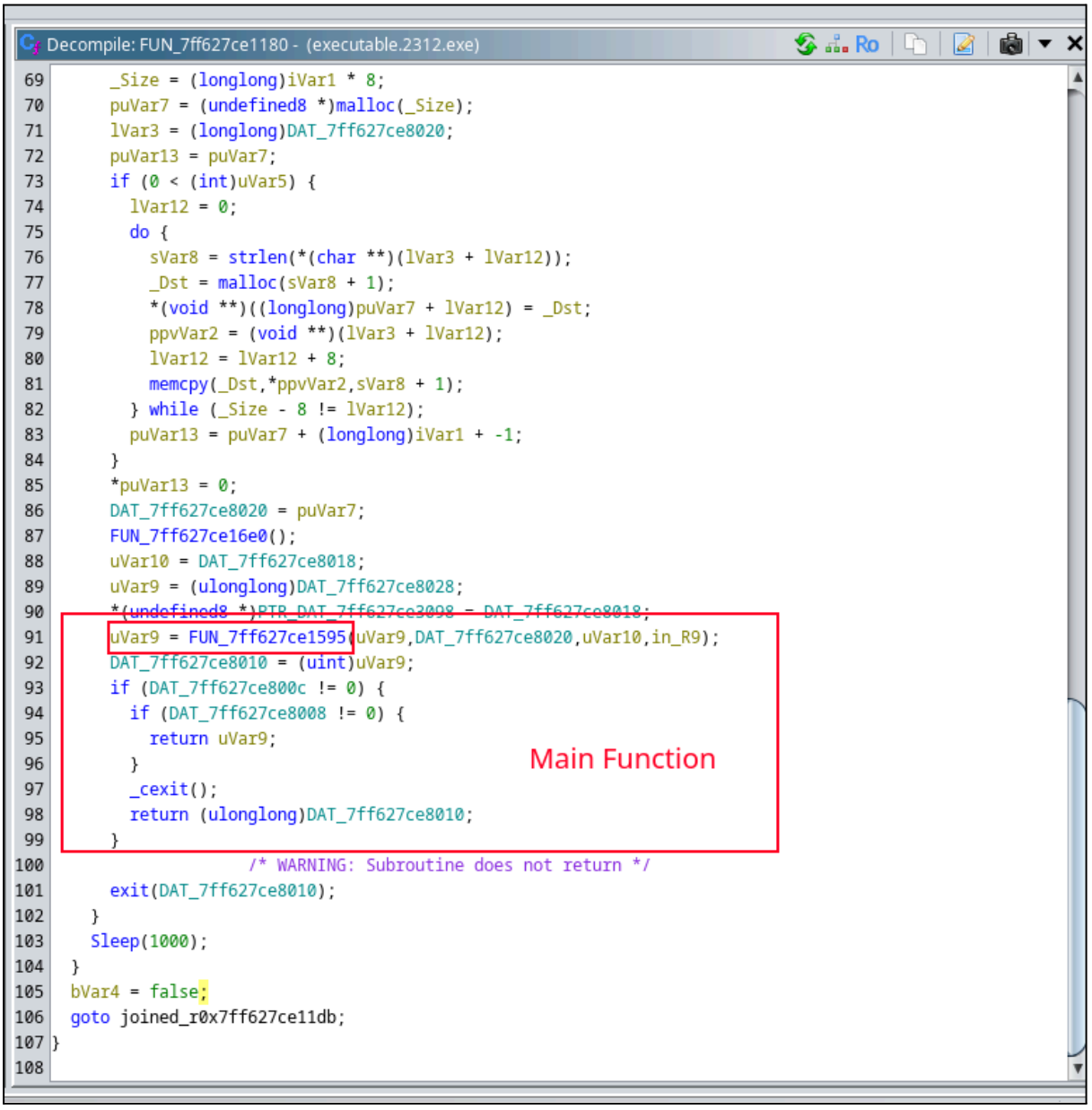


Figure 57: Entry point

A call to another method that had a string, a hexadecimal number, and two extra parameters was made inside the main function. The final piece of the flag, "**ith_xor}**" was revealed once the hexadecimal data was extracted and XORed with a certain key. The full flag was recreated as a result of deeper research of this function and its supporting rationale.



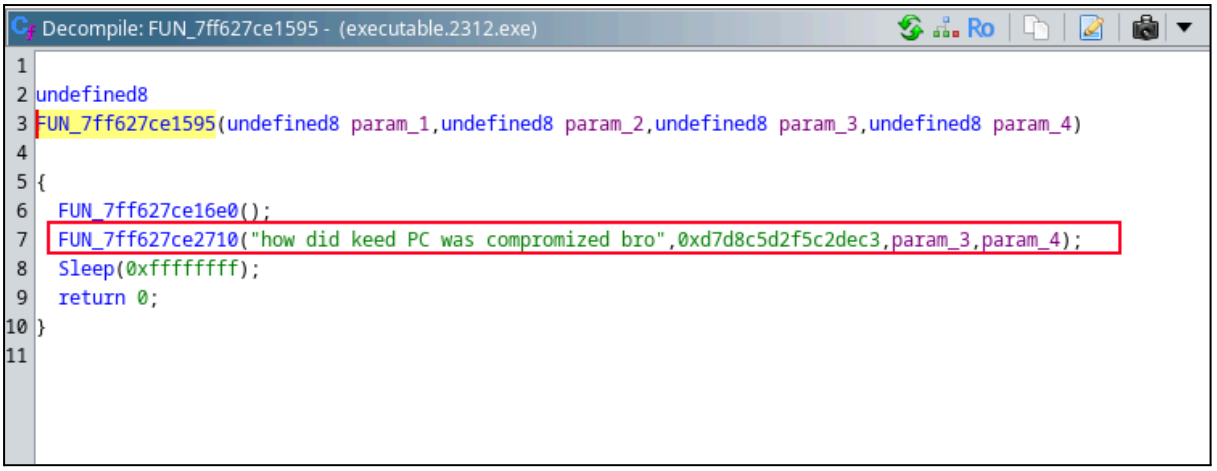


Figure 59: Main Function

The function's purpose was better understood after extracting the whole hexadecimal sequence after more analysis. At first, it looked like the output was a muddled signal. Nevertheless, the flag was successfully recreated in its original form following the accurate decoding and rearrangement of the hex data.

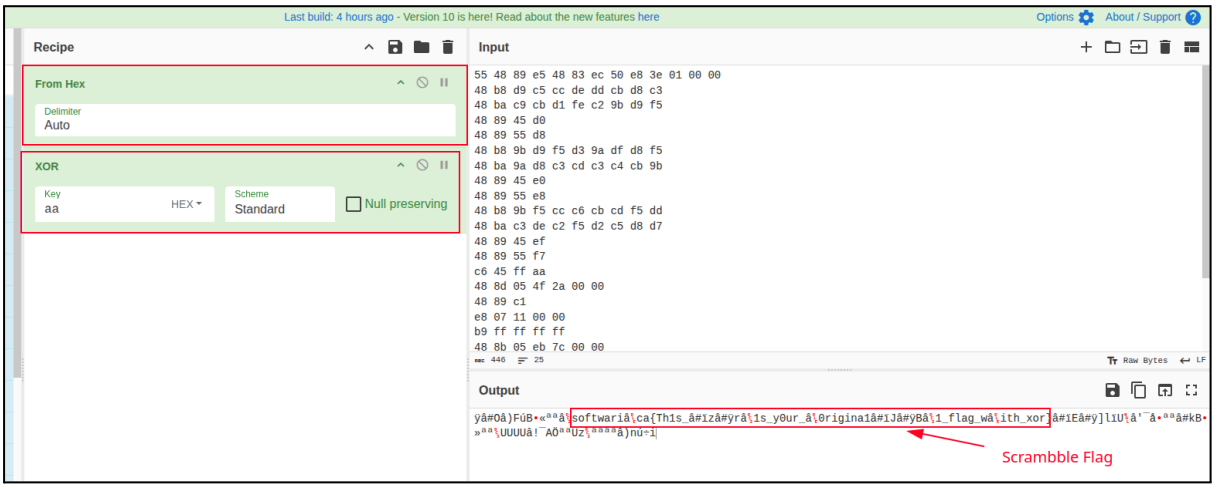


Figure 60: Scrabble Flag

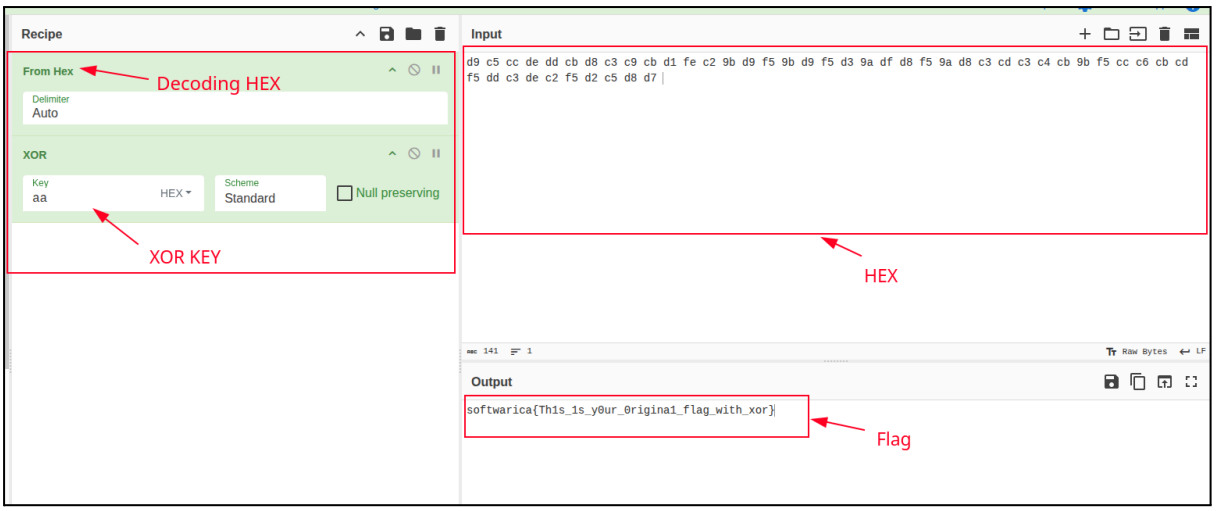


Figure 61: Fixed Flag

6.3. Summary of Findings

The memory forensic analysis focused on evaluating a volatile memory image to discover evidence of malicious activity taking place in-memory and to extract secret flags. Using grep and strings, the initial triage revealed flag-like patterns that contained the word "softwarica." Use of the Volatility framework was applied for more extensive investigation. The OS profile for proper plugin execution was discovered with the aid of the imageinfo plugin, and an analysis of the process tree turned up questionable child processes that signaled malicious action. Ghidra was deployed to do static reverse engineering after these processes were loaded for offline analysis. One executable was found to contain an embedded flag: save.exe (also known as executable.2312.exe). A call that carried out an XOR operation on hex-encoded data was made public by reverse engineering its primary function, which eventually divulged the flag's last element. Following data decoding and restructuring, the complete flag was effectively reconstructed. Even after a process has terminated, our research demonstrates how volatile memory can keep essential proof of secret payloads and attack actions.

7. MITRE ATT&CK Mapping of Observed TTPs

The following table maps the key adversary tactics, techniques, and sub-techniques observed during the Keed Loan Financial breach investigation to the MITRE ATT&CK® for Enterprise framework. (MITRE ATT&CK, 2025)

Domain	Key Findings	MITRE ATT&CK Tactics	Techniques (ID)	Evidence / Example	Conclusion
Email Forensics	- Attack initiated via phishing emails with obfuscated, corrupted PDFs and hidden links - Multi-stage delivery using chained attachments & ciphered clues - Used metadata and external services (Google Drive, Pastebin)	Initial Access, Defense Evasion, Collection, Command & Control	Phishing (T1566), Obfuscated Files (T1027), Encrypted Channel (T1573), Data from Info Repositories (T1213), App Layer Protocol (T1071)	Threatening email, base64-encoded malformed PDF, hidden Google Drive link, video flag, ciphered text in metadata	Sophisticated use of technical and social methods. Layered clues and obfuscation increased complexity, but disciplined forensic analysis traced the attacker's chain of evidence.
Network Forensics	- FTP server misconfiguration allowed anonymous access - LFI, file upload, SQL injection vulnerabilities exploited for lateral movement and persistence - Exfiltration over unencrypted channels & C2 traffic - Downloaded and scanned malware (e.g., Amadey, Mirai)	Initial Access, Execution, Persistence, Lateral Movement, Exfiltration, Impact	Exploit Public-Facing App (T1190), Valid Accounts (T1078), Command-line Interface (T1059), Web Shell (T1100), Remote File Copy (T1105), Exfiltrate over Web Service (T1567), Data Destruction (T1485)	Anonymous FTP login, evidence of file downloads, web shell upload attempts, cracked zip files, malware hash matches	Classic infrastructure weaknesses were exploited in a chain for access, persistence and impact. The investigation required file, protocol, and binary analysis across multiple network layers.
Memory Forensics	- Memory capture revealed volatile processes with flags and indicators - Suspicious processes (ransomware.exe, save.exe, etc) dumped and reverse-engineered - Recovered base64-encoded flags and analyzed malicious logic	Execution, Defense Evasion, Credential Access, Impact	In-Memory Execution (T1055), Indicator Removal (T1070), Obfuscated Files (T1027), Credentials in Memory (T1552/3), Data Destruction (T1485), Ransomware (T1486)	Volatility analysis, process tree showing malicious executables, Ghidra decompilation, recovered flags in RAM	Critical evidence was only present in volatile memory, demonstrating the necessity of RAM capture and advanced memory inspection to uncover in-memory malware and anti-forensic behavior.

8. Conclusion and Recommendations

A study of the Keed Loan Financial data breach shows a complicated, multi-phase intrusion that took advantage of technical weaknesses, smart evasion, and human manipulation. Attackers were able to get past core email security safeguards and acquire initial access by utilizing a targeted phishing email with hidden links and obfuscated attachments. The investigation demonstrated how the attackers purposely made it more difficult to detect and monitor their activity through a succession of evidence and clues that were implanted.

Vulnerabilities permitting file upload, SQL injection, exploitable LFI, and anonymous FTP access were identified using network forensics. These vulnerabilities allowed attackers to steal credentials, move laterally inside the environment, install sophisticated malware like Amadey and Mirai, and exfiltrate confidential client and company information. Inadequate patch management and network monitoring encouraged several steps of exploitation, persistence, and exfiltration, according to a timetable pieced together from logs and captures.

In-memory, fileless malware and hidden flags were found thanks to memory forensics, which also highlighted how the attackers exploited persistence and anti-forensic measures to evade disk-based detection. Using tools like Volatility and Ghidra, the team was able to analyze suspicious processes, extract malicious logic, and end the evidence chain, confirming the significance of RAM analysis in contemporary breach investigations.

The assault closely duplicated well-known MITRE ATT&CK techniques, including phishing, privilege escalation, ransomware deployment, and defensive evasion. The attacker's plan and impact could only be fully reconstructed via a multi-layered, multidisciplinary forensic technique.

Remediation: Keed Loan Financial should introduce stringent patch management procedures to quickly fix vulnerabilities, improve email filtering and user awareness training to lessen phishing threats, and use sophisticated endpoint detection and response (EDR) tools with real-time monitoring in order to reduce risks and stop future breaches. To further improve defenses, network segmentation, tightening access rules, restricting anonymous FTP access, and demanding robust multi-factor authentication (MFA) will be introduced. To find fileless

malware and advanced persistent threats early, the incident reaction plan should entail regular memory forensics and behavioral analysis.

References

Bluevoyant. (2022, March 8). *Understanding Digital Forensics: Process, Techniques, and Tools*. https://www.bluevoyant.com/knowledge-center/understanding-digital-forensics-process-techniques-and-tools

Dcodefr. (2015, May 13). *Cipher identifier*. dCode - Solveurs, Crypto, Maths, Codes, Calculs en Ligne. https://www.dcode.fr/cipher-identifier

Dubey, H., Bhatt, S., & Negi, L. (2023, July 4). *Digital Forensics Techniques and Trends*. The International Arab Journal of Information Technology. https://www.iajit.org/upload/files/Digital-Forensics-Techniques-and-Trends-A-Review.pdf

Fox, N. (2023, April 6). *How to use Ghidra to reverse engineer malware | Varonis*. Varonis: We Protect Data. https://www.varonis.com/blog/how-to-use-ghidra

Gupta, S., Gupta, S. C., Gupta, M., & Agarwal, A. (2011, January 17). *Systematic Digital Forensic Investigation Model*. Researchgate. https://www.researchgate.net/profile/Yatendra-Gupta/publication/228410430_Systematic_Digital_Forensic_Investigation_Model/links/56ea8cd208ae95bddc2bcc6b/Systematic-Digital-Forensic-Investigation-Model.pdf

Harvey, P. (2025, July 25). *ExifTool*. ExifTool by Phil Harvey. https://exiftool.org/

Ieong, R. S. (2006, September 14). *Digital forensics investigation framework that incorporate legal issues*. Forza. https://www.sciencedirect.com/science/article/pii/S1742287606000661

Jekyll. (2024, March 22). *Volatility guide*. jloh's Blog. https://blog.jloh02.dev/ctf/volatility-guide/

Jody-admin. (2025, February 25). *Cyberchef tutorial and tips*. HackerTarget.com. https://hackertarget.com/cyberchef-tutorial-tips/

Linuxcommandlibrary. (2020, June 3). *Binwalk*. Linux Command Library. https://linuxcommandlibrary.com/man/binwalk

MITRE ATT&CK. (2025, April 22). *MITRE Framework*. MITRE ATT&CK®. https://attack.mitre.org/

Openwall. (2017, March 10). *John the Ripper password cracker*. Openwall - bringing security into open computing environments. https://www.openwall.com/john/

Paloalto. (2020, May 26). *What is an incident response plan getting started*. Palo Alto Networks. https://www.paloaltonetworks.com/cyberpedia/incident-response-plan

Virustotal. (2024, September 6). *How it works*. VirusTotal. https://docs.virustotal.com/docs/how-it-works

Wireshark. (2020, June 13). *Wireshark · Documentation*. https://www.wireshark.org/docs/

Appendix

Memory Fake Flags

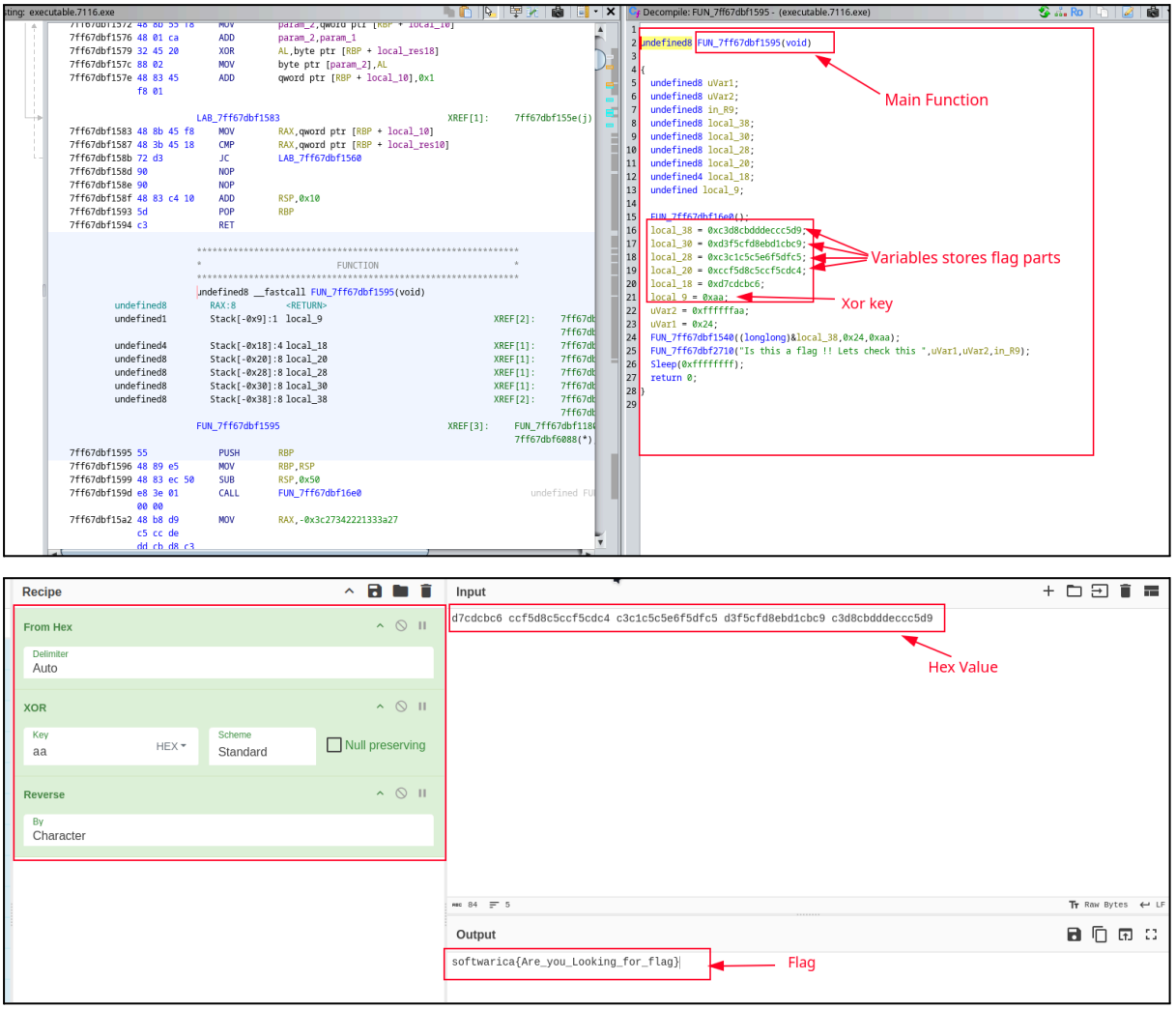
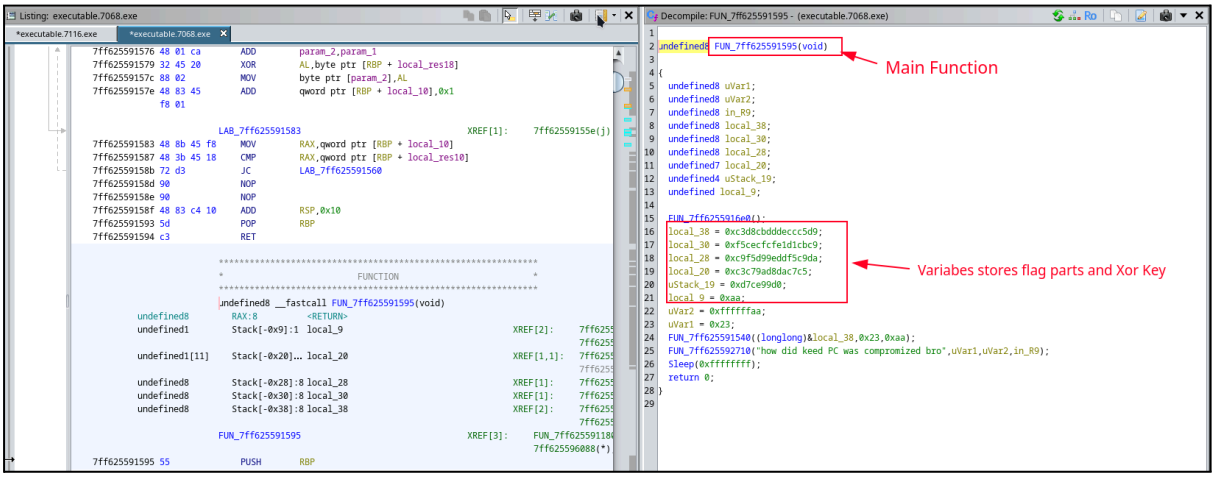


Figure 62: Ransomware.exe Flag



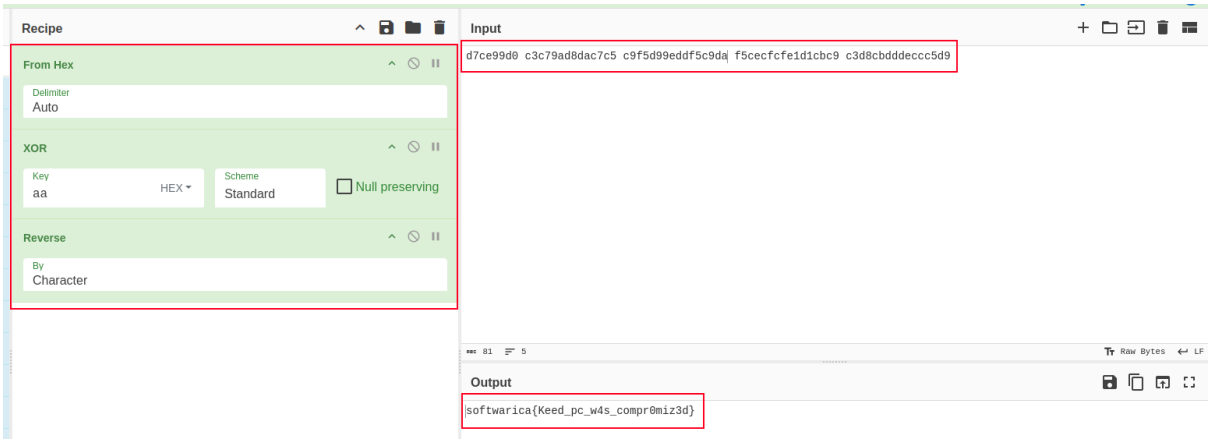


Figure 63: Malware.exe Flag

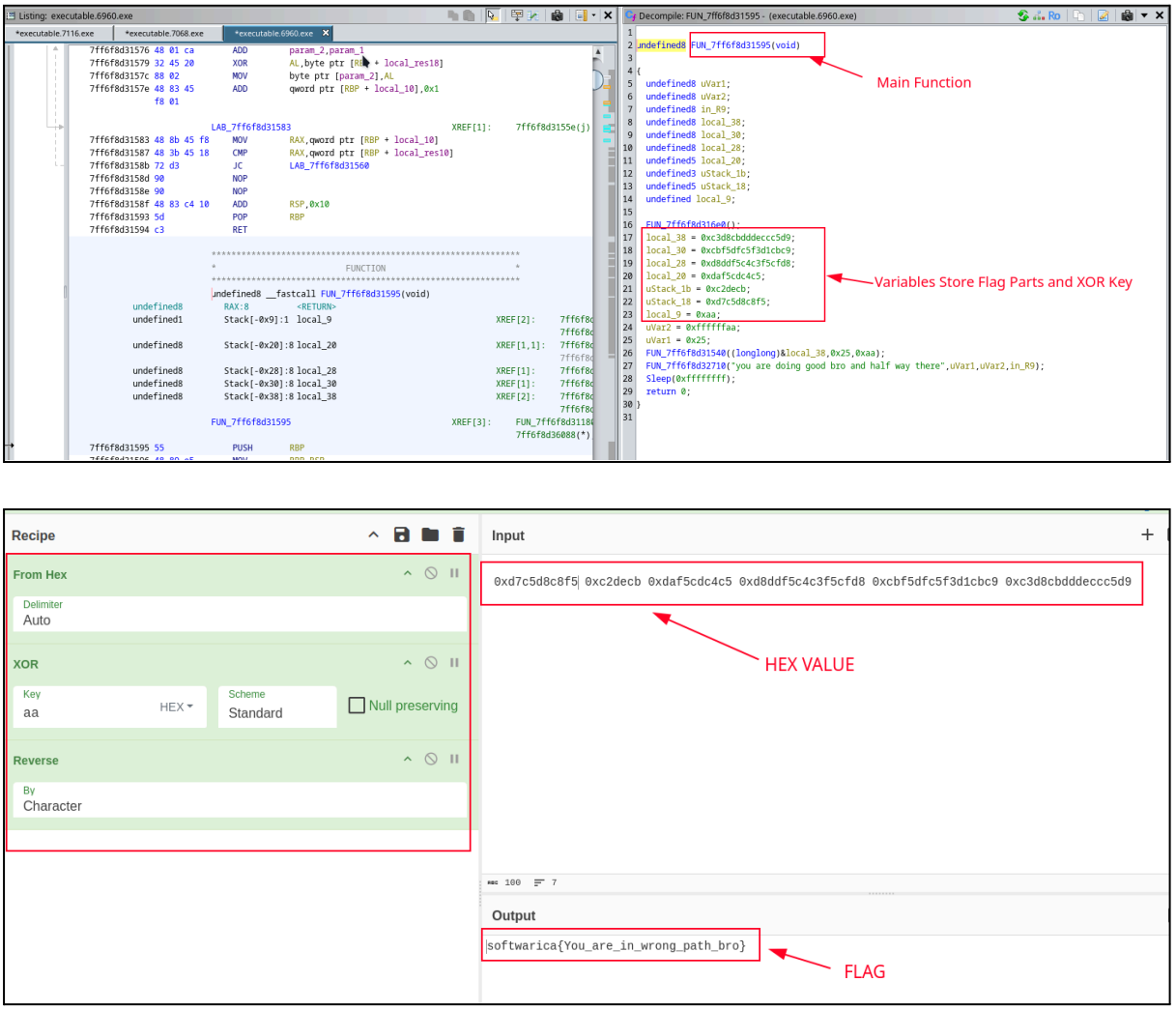


Figure 64: Flag.exe Flag