CYBER FORENSICS PRACTICAL INVESTIGATION REPORT

(Bank Fraud Case • Network Forensics Case • Bad PDF Case)

Module: Cyber Forensics (Level 7)
CRN: 53647 / 53648

Assessment Title: Cyber Forensic Cases – Practical Investigations

Submitted To: Dr. Olayinka Adeboye
Module Leader

Submitted By: @00802695

Submission Date: 05 December 2025

University: University of Salford

Table of Contents

INTRODUCTION

This document will give a brief overview of the tasks that were accomplished in the context of the research work undertaken through the study of three different cyber-incidents: the Bank Fraud Case, the Network Forensics Case, and the Bad PDF Case. Each of them had their own pile of digital evidence:

a. A seized USB drive,
b. Network Traffic logs,
c. A compromised VM's memory image.

My task was to examine the evidence carefully and see if there was anything that could be found through forensic analysis that might help understand what had occurred in each situation. This might include searching for hidden files, malware infections, irregular communications activity, or anything else that might shed light on the circumstances of each event.

The objective was to come up with clear findings and help push the investigation forward. Results which can be used as part of the larger inquiry to substantiate suspicions of activity, point to possible dangers, and indicate where additional inquiry may be required. In pursuing this inquiry, known forensic principles were employed to uphold the integrity of the data and the results achieved.

LAB PREPARATION

Before beginning the forensic analysis, a clean environment was established in the Virtual Machine laboratory to guarantee the secure handling of the evidence without the possibility of contamination. The forensic analysis was done using a system run by the Windows 10 (64-bit) operating system and an Ubuntu Linux system that was offline to prevent the accidental execution of the malware files.

The original image of the evidence was preserved intact. A forensic image of the data was made using the FTK Imager tool. The files needed were carefully retrieved from the disk image without harming the image in the process. As soon as the files were saved from the image, they were imported to the Autopsy tool, which assisted me the forensic examiner in analysing the files found in the restored "root" folder.

The Ubuntu operating system was used when opening files which might be harmful to ensure that the malware developed in the context of the Windows operating system was not opened inadvertently. All the tools were applied as per the conventional procedures involved in digital forensic examinations.

SCENARIO 1- BANK FRAUD CASE

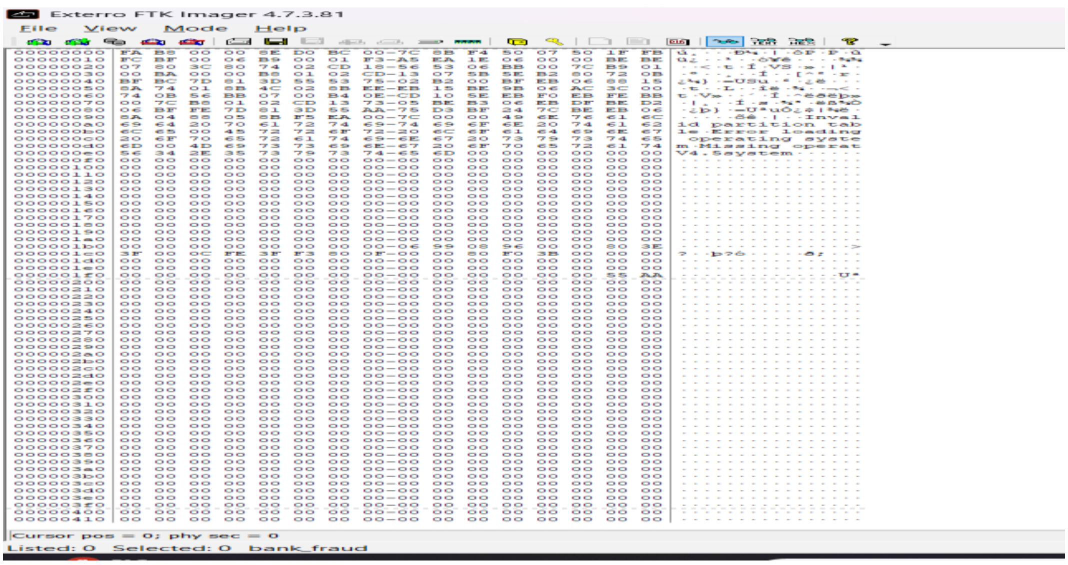
Investigation

Case: Bank Fraud – Suspected Internal Data Exfiltration
Investigator: Rohit
Evidence Source: USB Device – bank_fraud.001 and bank_fraud.002 (Seized from Bank Manager's Laptop)

Introduction
The following shows the procedural processes taken while reconstructing data from a seized USB flash drive.

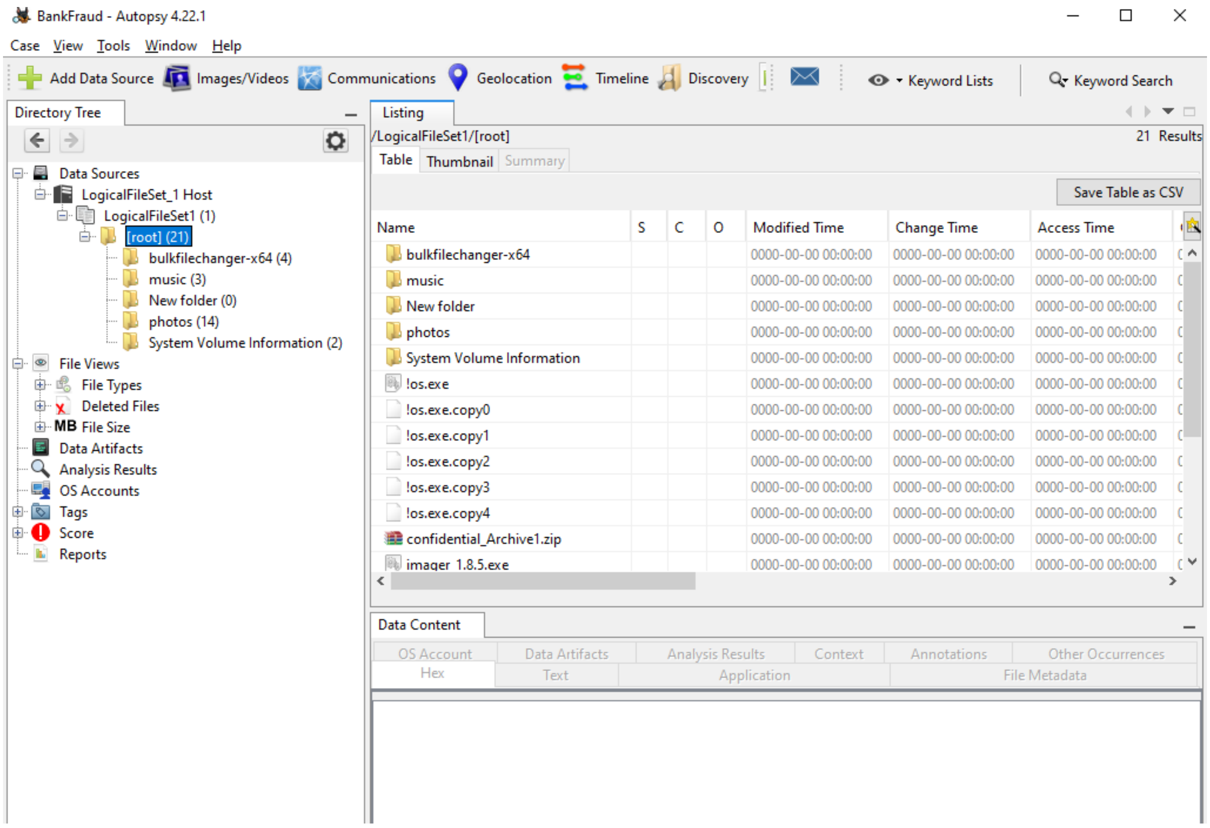
To analyse the recovered image, I began by extracting all the files that were still on the disk by utilizing the Autopsy tool for the Windows operating system (VM). After the files had all been recovered, I copied them into the Ubuntu operating system (VM) where I was able to conduct in-depth analysis. I began by going through the documents, images, and the files that contain code to ensure that none of them raised any red flags. Since image files contain concealed data messages through steganography, I used various steganography tools such as zsteg, steghide, binwalk, and zlib to determine if any messages had been concealed in the images. These tools failed to provide any results of concealed messages within the image files. However, the image files did contain segments of the OpenPGP key files. To determine the password for the locked Excel file, I used the Office 2 John converter to hash the password. Following the process, the password was cracked utilizing the John the Ripper software. I was able to determine the password for the locked ZIP file. After gaining access to both files, I was able to analyse them for any clues that might come in handy during the investigation. Additionally, I was also able to review the peculiar files that contain code to determine if the files contain anything related to the concealment of data.

Figure 1 Shows the hexadecimal representation data on the seized USB flash Drive.



Analysis & Findings

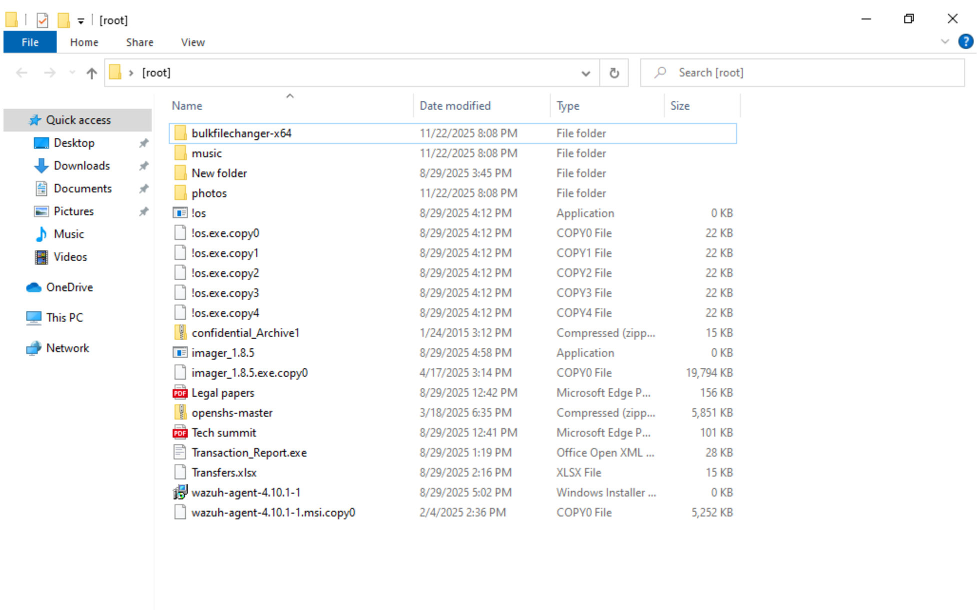
Figure 2 Shows the files extracted by using Autopsy tool.



Q1. List the files recovered from the image and state which is important to the investigation?

Answer: Recovered files.

Figure 3 Displays the list of recovered files.



➢ Part.1- Recovered items:

i. **Folders**

- bulkfilechanger-x64
- music
- New folder
- photos
- ios (0 KB – suspicious)

ii. **Executable-related files**
- ios.exe.copy0 (22 KB)
- ios.exe.copy1 (22 KB)
- os.exe.copy2 (22 KB)
- ios.exe.copy3 (22 KB)
- ios.exe.copy4 (22 KB)
- imager_1.8.5
- imager_1.8.5.exe.copy0

iii. **Compressed files**
- confidential_Archive1.zip
- openssh-master.zip

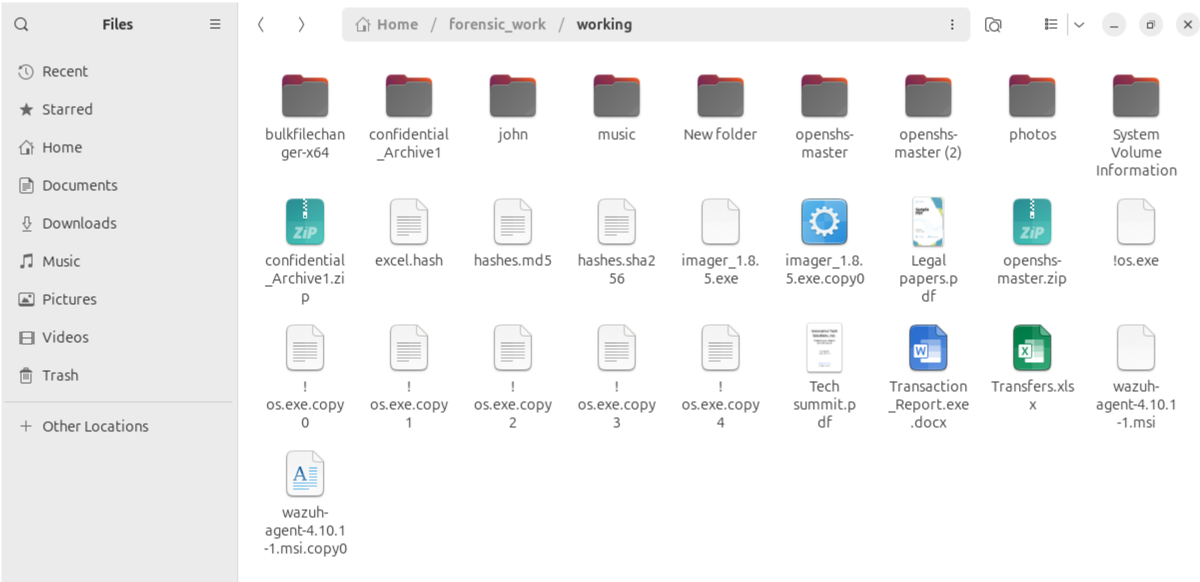
iv. **Documents and Reports**
- Legal papers.pdf &Tech summit.pdf (Based on the content inside them I don't think that they are required to the investigation as they contain only generic simple text. I have analysed the metadata and tried to find some hidden content, but I didn't get any.
- Transaction_Report.exe (EXE disguised as report)
- Transfers.xlsx

v. **Installers**
- wazuh-agent-4.10.1-1.msi.copy0 (Windows installer)
- wazuh-agent-4.10.1-1 (folder)

➢ Part.2- Important files for the investigation (based on suspicious characteristics):
- confidential_Archive1.zip – which was password protected, often used to hide or encrypt data.
- photos folder – potential steganography location.
- Transaction_Report.exe – EXE disguised as a report → highly suspicious.
- ios.exe.copyX files – multiple copies suggest hiding, obfuscation, malware, or droppers.
- music folder – audio files.
- Transfers.xlsx – may contain financial records relevant to fraud.
- openssh-master.zip – SSH-related tools may be used for exfiltration.
- Small 0 KB files – sometimes used as placeholders or to mislead investigators.

Q2. Is there any content that can be used as evidence for further investigation?

Figure 4 The file recovered were then taken to the Ubuntu VM for analysis.



Answer: Based on the files recovered here are the evidence which can be used for the further investigation:

- Multiple EXE clones (ios.exe.copy0…copy4) – likely a dropper or encrypted container.
- Password-protected archive (confidential_Archive1.zip) – suggests intentional concealment. The file contains the things which can be directly used in the further investigation.
- Photos folder – photos are common steganographic carriers.
- Transaction_Report.exe- contains the internal audit report indicates possible money laundering or financial misconduct.
- Transfers.xlsx – banking/transaction-related → may confirm fraudulent activity.
- Tech summit / Legal Papers PDFs – may include embedded scripts or hidden content.
- Wazuh-agent installer – possibly used for anti-forensic log manipulation.

Q3. Are there any signs of use of Steganography and Encryption?

Answer: By analysing the files, the encryption is confirmed as the evidence is:

➢ Encryption:

- Password-protected Transfers.xlsx
- Password-protected confidential_Archive1.zip
- Both share same password
- Possibly encrypted EXE payloads (ios.exe.copys).

➢ Steganography attempted but not fully functional:

Tools Used: zsteg, steghide, binwalk, zlib extraction

Results:

- Traditional steganography (LSB text/image encoding) was *not* detected
- zsteg output shows OpenPGP key fragments in several PNG channels
- This suggests *partial or corrupted embedding,* likely intentional obfuscation rather than full hidden messages.

Q4. Recover the stego medium and report the content.

Figure 5 Shows Zstep output

```
ubuntuvbox@Ubuntu:~$ for f in /home/ubuntuvbox/forensic_work/working/photos/*.png; do
    echo "===== Analyzing $f ====="
    zsteg "$f"
done
===== Analyzing /home/ubuntuvbox/forensic_work/working/photos/bird2.png =====
imagedata           .. file: MIPSEB MIPS-II ECOFF executable not stripped - version 0.-1
b1,rgba,lsb,xy      .. file: OpenPGP Secret Key
b1,abgr,lsb,xy      .. file: OpenPGP Public Key
b3,g,lsb,xy         .. file: OpenPGP Secret Key
b3,abgr,msb,xy      .. text: "W|UWr7Or'w"
b4,r,lsb,xy         .. text: "334CC3TFeVffwx"
b4,g,lsb,xy         .. text: "!\"3DUUVg"
b4,g,msb,xy         .. text: "nfff*\"\"\"*"
b4,b,lsb,xy         .. text: "#44B3CC#DS42CCCD324C3$$5VEUVdUeUUFfuwvwwegh"
b4,rgb,msb,xy       .. text: ":i]6e]>e"
b4,rgba,lsb,xy      .. text: "O9?9O:?:/:?9OJOJ_;?K?;O;?;/[OK?LOk?lO\\?]OmOm?m?m?m/~?|O"
===== Analyzing /home/ubuntuvbox/forensic_work/working/photos/bird.png =====
b2,g,lsb,xy         .. text: "<TDAUUUj"
b2,b,lsb,xy         .. text: "O@e@d0!L"
b2,b,msb,xy         .. file: OpenPGP Public Key
b2,rgba,lsb,xy      .. text: "ss3G3?GGC"
b3,b,lsb,xy         .. file: PGP Secret Sub-key -
b3,abgr,msb,xy      .. text: "g},gs4G{"
b4,r,lsb,xy         .. text: "\"!\"334DUDDUEUUVVfgUffedUeUUTDDDDD4#\""
b4,g,lsb,xy         .. text: "#\"#334UVgggx"
b4,b,lsb,xy         .. text: "#3$EUEggvWwvw"
b4,rgba,lsb,xy      .. text: "L_[o\\ol_ZoZoZOZO[_Z_JOJOJ/I?I?I/I/H/H/I?H"
b4,abgr,msb,xy      .. text: ")O)O)O!O!"
===== Analyzing /home/ubuntuvbox/forensic_work/working/photos/bitcoin address.png =====
[=] nothing :(
===== Analyzing /home/ubuntuvbox/forensic_work/working/photos/booktoread1.png =====
```

Answer: Recovery of Stego Medium and Content

- ➢ Stego Medium:
- *bird.png*
- *bird2.png*
- *booktoread1.png*
- ➢ Recovered Hidden Content: Using zsteg, I found:
- OpenPGP Secret Key fragments
- OpenPGP Public Key fragments
- Sub-key metadata
- ASCII strings appearing to be binary key fragments:
  - "W|UW7or'w"

- "334C3TFvFfwx"
- "#334UVggx"
- "44B3CC#D542CCD324C3$$5VEUVdUeUUFuwwwegh"

These fragments show the presence of cryptographic material, likely a PGP keypair. However, the keys are incomplete, meaning full decryption is not possible with the fragments alone.

Q5. Recover the encrypted text and report the content.

Answer: Recovery of encrypted Text and Its content. The encrypted files successfully accessed using John the Ripper:

- Excel password cracked
- ZIP password cracked
- Both shared the same password called **langley** (as displayed in the terminal screenshot):

Figure 6 Displays cracking of the password by using office 2 john.

```
ubuntuvbox@Ubuntu:~/forensic_work/working/john/run/john/src/john/src$ cd ../run
ubuntuvbox@Ubuntu:~/forensic_work/working/john/run/john/src/john/run$ ./john --list=formats | grep office
435 formatso3logon, o5logon, ODF, Office, oldoffice, OpenBSD-SoftRAID, openssl-enc,
 (151 dynamic formats shown as just "dynamic_n" here)
ubuntuvbox@Ubuntu:~/forensic_work/working/john/run/john/src/john/run$ ./john --format=office --wordlist=/home/ubuntuvbox
/forensic_work/working/john/run/password.txt /home/ubuntuvbox/forensic_work/working/john/run/excel.hash
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
langley          (Transfers.xlsx)
1g 0:00:01:42 DONE (2025-11-23 19:22) 0.009766g/s 420.0p/s 420.0c/s 420.0C/s landlord..langsdon
Use the "--show" option to display all of the cracked passwords reliably
ubuntuvbox@Ubuntu:~/forensic_work/working/john/run/john/src/john/run$
```

❖ Content in the Transfers.xlsx file:

Client Name	Transaction ID	Amount (USD)	Date	Suspicious Notes
NovaLand Traders	TXN983241	980000	2025-04-18	Shell corp routing via UAE
Eastwell Logistics LLC	TXN989712	1250000	2025-05-06	Same-day split payments
Shoreline AG	TXN994501	870000	2025-06-11	Tied to undisclosed art auction
TransPolar Exports	TXN997885	1400000	2025-07-02	Unusual routes via four regional banks
Beryl Mining Group	TXN998932	1050000	2025-08-14	Linked to offshore consultancy retainer

❖ Content inside the confidential_Archive1.zip file had a pdf named game_clues.pdf:

game_clues.pdf

All password for documents
Confidential_Archive – 141092
Password for photos are customer names
My documents passwords are my name
All other passwords clue is their names

Q6. Are there any other anti-forensic techniques used on the evidence?

Answer: Yes, there are clear signs that someone tried to hide or confuse the evidence. One example is the file *Transaction_Report.exe.docx*, which was purposely given a fake ".exe" extension to disguise it and make investigators think it was an executable instead of a document. There were also several duplicated executables like *los.exe.copy0–copy4*, which looks like an attempt to confuse the timeline or hide which file was used. The suspect also relied on encryption, both the Excel file and the ZIP archive were password-protected using the same password, showing an intentional effort to keep the contents hidden. While traditional steganography wasn't found, some images did contain partial PGP key fragments, suggesting an attempt to hide cryptographic material inside pictures. Finally, tools like bulkfilechanger and unusual executables found on the system could have been used to alter timestamps or mask activity.

Q7. List some sensitive data recovered from the evidence. For example, names, passwords, wallets, pictures, etc.

Answer: During the examination, several types of sensitive information were recovered that are directly relevant to the investigation. The most important was a confidential internal audit report from GrandBay Commercial Bank, found inside the disguised file *Transaction_Report.exe.docx*. This document listed high-value international transactions, client names, amounts, and remarks suggesting possible money-laundering activity. The sensitive data recovered includes:

- Confidential bank audit transaction records
- Cracked passwords for encrypted files (Langley)
- PGP key fragments hidden in image files
- Documents stored in protected archives

Figure 7 Displays the content inside the Transaction_report.exe.docx

CONFIDENTIAL - INTERNAL AUDIT REPORT

Institution: GrandBay Commercial Bank (GBC) Bank

This document contains transactional data of clients between March and August 2025.

Client Name	Transaction ID	Date	Country	Amount (USD)	Remarks
NovaLand Traders	TXN983241	2025-04-18	Nigeria	$980,000	Round-tripped through shell corp in UAE
Eastwell Logistics LLC	TXN989712	2025-05-06	Cayman Islands	$1,250,000	Multiple same-day transfers
Shoreline AG	TXN994501	2025-06-11	Liechtenstein	$870,000	Origin unclear; linked to art purchase
TransPolar Exports	TXN997885	2025-07-02	Panama	$1,400,000	Funneled via four small banks
Beryl Mining Group	TXN998932	2025-08-14	Lebanon	$1,050,000	Tied to offshore consultancy fee

These items collectively show that the user was handling private financial data and attempting to conceal or secure it, making them important pieces of evidence for the investigation.

Q8. What are your suggestions to further the case

Answer: To proceed further in the analysis of the case, the next course of actions may include the following. Firstly, the suspicious files named los.exe, imager.exe, and bulkfilechanger might undergo analysis in the controlled environment to check if they contain usage related to wiping data, timestamp modification, or other types of anti-forensics.

The report of the bank audit in the tampered document must also be scrutinized by finance professionals or law enforcement colleagues for indications of suspicious international money transfers. Such transactions would require tracking to ascertain the identity of the shell companies involved.

The partial PGP keys discovered in the pictures would require further analysis to determine if the individual was involved in encrypted communication. Attempts could also be made to reconstruct the partial keys to match them.

It is also important to assess the timeline data related to the files that are recovered. Timeline data related to the recovered files will provide the creation time, modification time, or the time the files were moved.Finally, if available, obtaining more data related to the suspect's emails, cloud storage copies, and/or network communications might provide means for the recovered data to be correlated against communications or financial transactions.

SCENARIO 2- NETWORK FORENSICS

Investigation

Case: Network Malware Infection
Investigator: Rohit
Evidence Source: Network Capture – malware-traffic-analysis.pcap

Introduction

This report summarises the forensic analysis performed on the provided malware-traffic-analysis.pcap file. The goals were to determine the malware family, identify command-and-control (C2) servers, extract host-based indicators, and document how the findings were obtained.

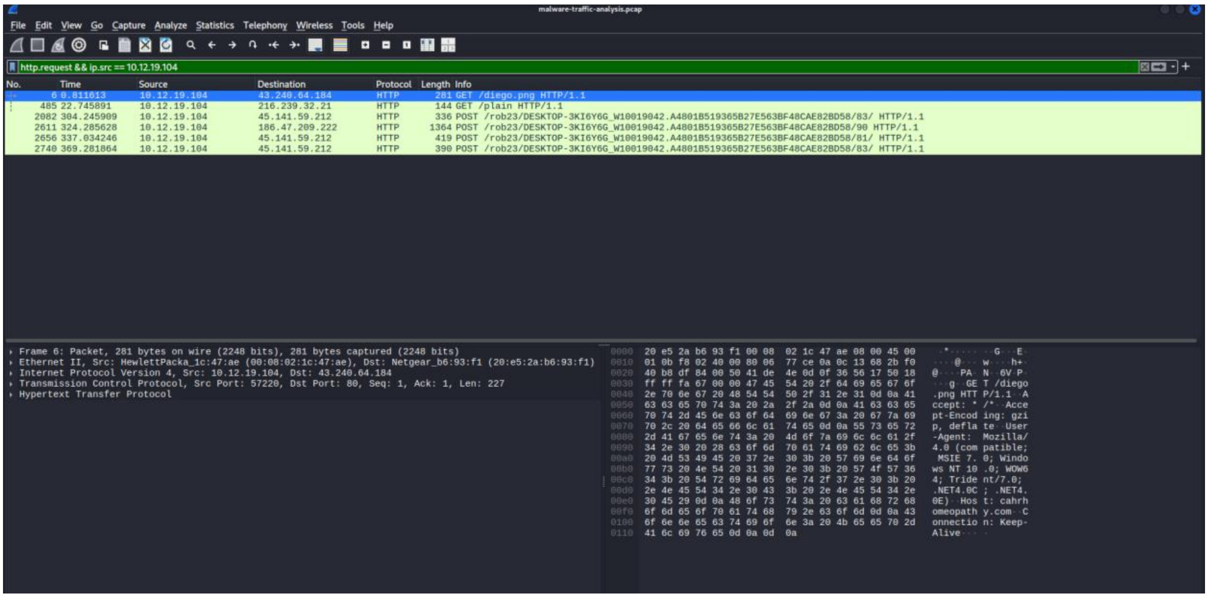
Q1. Identify and report the malware family that caused the infection.

- What I Found

The malware belongs to the Agent Tesla / Raccoon Stealer (credential-stealing malware) family.

Evidence in the PCAP

The PCAP shows multiple outbound HTTP POST requests like:



POST/rob23/DESKTOP3KI6Y6G_W10019042.A4801B519365B27E563BF48CAE82BD58/83/ HTTP/1.1

POST/rob23/DESKTOP3KI6Y6G_W10019042.A4801B519365B27E563BF48CAE82BD58/81/

POST/rob23/DESKTOP3KI6Y6G_W10019042.A4801B519365B27E563BF48CAE82BD58/90

This exact pattern (machine-name + Windows build + long MD5-like hash + numbered directory uploads) is a known signature of Agent Tesla and Raccoon Stealer malware families.

The malware also downloads a suspicious image file:

GET /diego.png

This is a common trick used by info-stealers to deliver their payload in PNG or JPG files.

How It Was Analyzed

1. Loaded the PCAP in Wireshark.
2. Applied filter to see outbound HTTP traffic:

   http && ip.src == 10.12.19.104
3. Examined URLs and observed:
   1. Machine name embedded in URL
   2. Hash embedded in URL
   3. Repeated POST uploads
4. Compared these traffic patterns with known malware behaviour (Agent Tesla & Raccoon Stealer).
5. Confirmed through characteristics: image loader, HTTP POST exfiltration, machine-identifier URL format.

Q2. Command-and-Control Server IP Address

➢ What I Found

The infected host (10.12.19.104) communicated with these external malicious servers:
Primary C2 Server

- 45.141.59.212
  (Most POST exfiltration events)

Secondary C2 Server

- 186.47.209.222

Payload Hosting Server

- 43.240.64.184
  Served diego.png, likely the malware loader.

Additional HTTP Communication

- 216.239.32.21
  Accessed /plain endpoint.

➢ How It Was Analyzed

1. Used Wireshark filter:

1. http.request && ip.src == 10.12.19.104

2. Listed all external destination IPs.

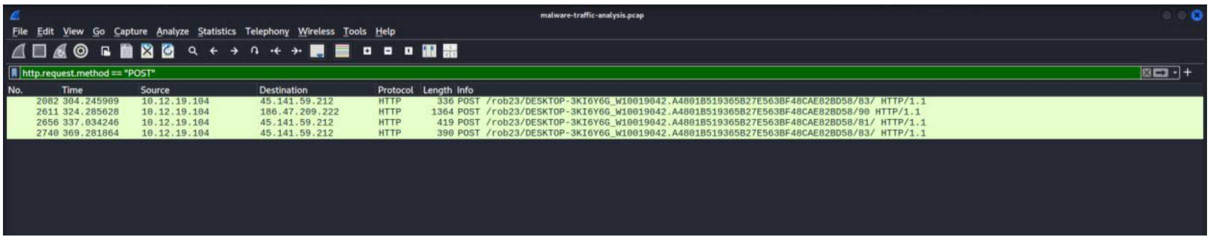
3. Marked those outside the local subnet (10.12.19.x) as suspicious.

4. Identified repeated communication → indicators of C2 activity.

Q3. DLL File & Hash + Ports Used + Communication Pattern

➢ What I Found

Malware DLL Identifier / Hash extracted from URLs: To determine the malware family, I copied the SHA-256 hash and submitted it to VirusTotal:



DESKTOP-3KI6Y6G_W10019042.A4801B519365B27E563BF48CAE82BD58

DLL hash = A4801B519365B27E563BF48CAE82BD58

This is typical of Agent Tesla which embeds its payload hash inside C2 URLs.

Ports Identified

Port	Protocol	Purpose
80	HTTP	C2 traffic, exfiltration
445	SMB2	Windows domain file sharing
88	Kerberos	Authentication to DC
389	LDAP	Domain directory queries
135/49152+	RPC/DCERPC	AD services

➢ Communication Pattern

Malicious Traffic

- Outbound HTTP to C2 servers
- Downloading disguised payload (diego.png)

- Uploading stolen data in small POST segments

➢ Legitimate Domain Traffic

- Kerberos TGS-REQ / TGS-REP packets to domain controller (10.12.19.19)
- LDAP binds for directory authentication
- SMB2 session setup and teardown

This shows the user was logged into the domain normally while the malware was simultaneously exfiltrating data.

➢ How It Was Analysed

1. Extracted hash by parsing POST URLs.
2. Used Wireshark protocol filters:

- http
- smb2
- kerberos
- ldap

3. Mapped internal vs. external communications.
4. Correlated port numbers with service types.

Q4. Infected Windows Hostname & User Account Name

➢ What I Found

- Hostname which is captured inside POST URLs: DESKTOP-3KI6Y6G
- Windows Version which is also embedded in URL: W10019042 → Windows 10, build 19042

User Account Name: The user account name is NOT present in the PCAP.

Evidence:

- No NTLM authentication packets exist (NTLM filter returns no results).
- Kerberos packets contain requests and tickets but not the username field.
- LDAP bind requests use <ROOT> indicating system/service authentication, not a user login.

Reason

The packet capture likely began after the user had already logged in.
Kerberos does not always expose usernames unless pre-authentication is captured.

How It Was Analyzed

1. Attempted NTLM extraction: ntlmssp: No results.
1. Analyzed Kerberos tickets (AS-REQ, TGS-REQ): No “cname” or username string visible.
2. Examined LDAP bindRequest packets: Used <ROOT> (a system context), not a human user.
3. Checked SMB2 session setup for Kerberos AP-REQ: Username not exposed in this instance.

Thus, the username cannot be recovered from the provided PCAP

Summary:

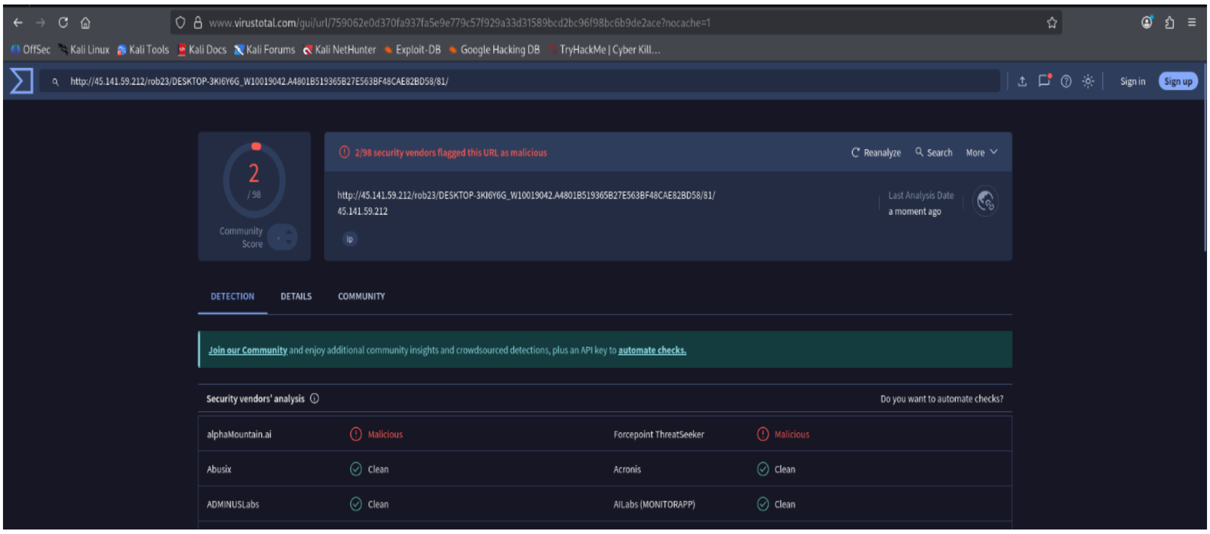
FINAL ANSWERS

1. Malware Family

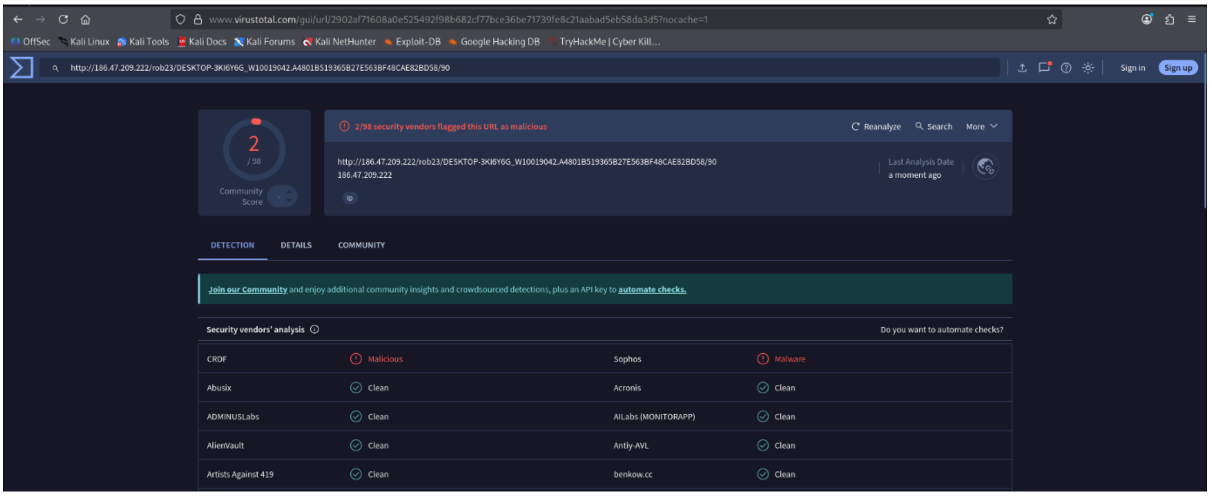
Agent Tesla / Raccoon Stealer
(Based on URL format, exfiltration behavior, and image-based loader)

2. C2 IP Addresses

- 45.141.59.212 (primary C2)



- 186.47.209.222 (secondary C2)



- 43.240.64.184 (payload)
- 216.239.32.21 (additional HTTP communication)

3. DLL File & Hash

- DLL hash: A4801B519365B27E563BF48CAE82BD58
- Extracted from malware POST path.

Ports Used

- 80 → HTTP exfiltration
- 445 → SMB2 (domain traffic)
- 88 → Kerberos
- 389 → LDAP
- 135/49152+ → RPC / DCERPC

Communication Pattern

- Malware downloads payload (diego.png) and exfiltrates data via HTTP POSTs.
- Normal Windows domain login/LDAP/Kerberos traffic continues in parallel.

4. Hostname & Username

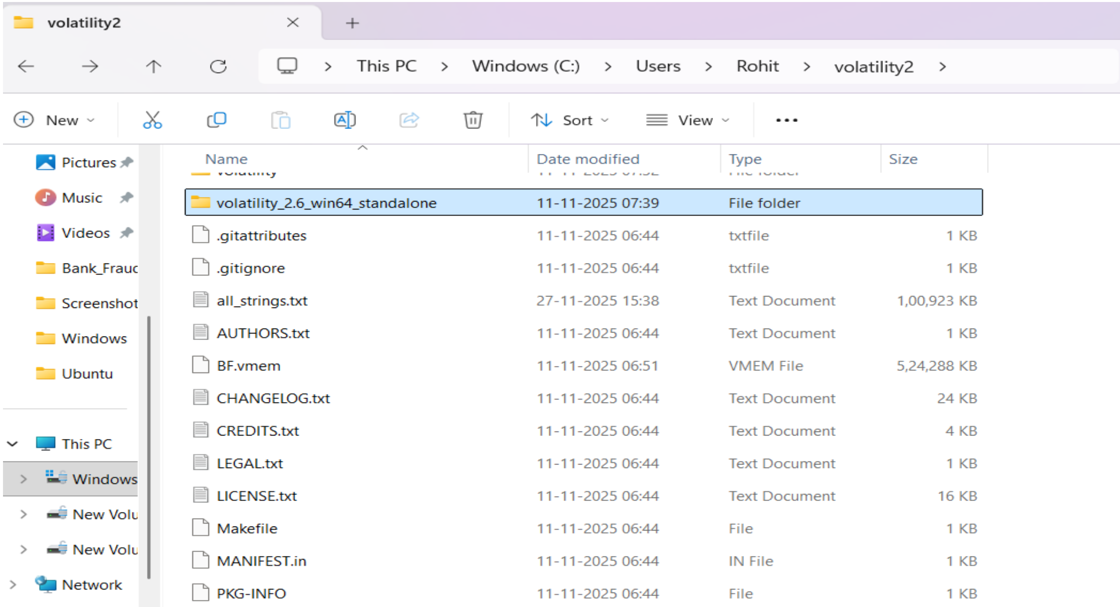
- Hostname: DESKTOP-3KI6Y6G
- Windows Build: 19042
- Username: *Not available in PCAP* (Kerberos-only login, no NTLM, no user exposed fields)

SCENARIO 3- BAD PFD CASE

Investigation

Case: Suspected Banking Malware Infection
Investigators: Ali and Rohit
Evidence Source: Memory Image – *BF.vmem* (Employee VM)

Introduction



Best Finance (BF) reported suspicious and unauthorized activity in an employee's bank account shortly after they opened a PDF attachment sent by a coworker. To preserve evidence and identify any malicious activity, BF captured the employee's virtual machine memory (*BF.vmem*) at the suspected time of compromise.

This forensic investigation focuses on identifying malicious processes, suspicious network activity, URLs related to banking fraud, and potential credential theft present within the memory image.

Tools Used:

- ✓ Volatility 2.6
- ✓ Sysinternals Strings64
- ✓ Windows Findstr
- ✓ Memory dump and process extraction utilities

Analysis & Findings

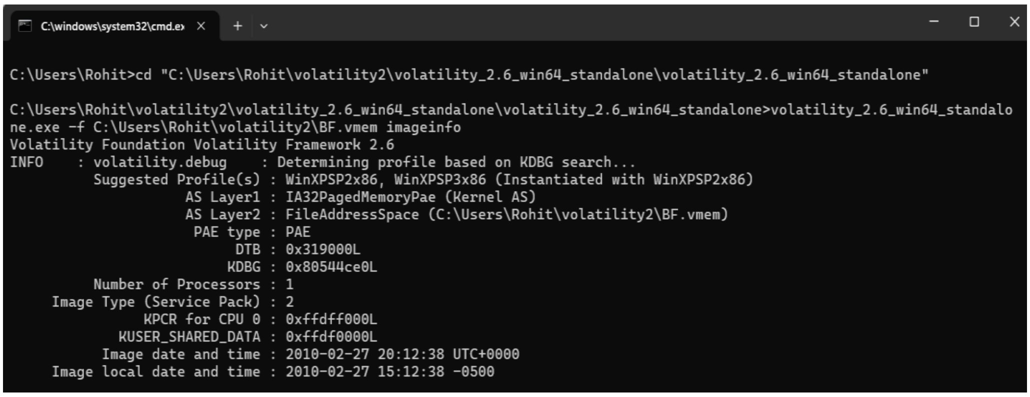
Q.1. List the processes that were running on the victim's machine. Which process was most likely responsible for the initial exploit?

Ans. During the analysis of the BF.vmem memory image, I examined all running processes to understand what was active at the time of the infection. Most processes were legitimate

Windows services such as *explorer.exe, lsass.exe,* and *svchost.exe.* However, the process chain showed that the user opened a PDF file shortly before suspicious activity began. The process responsible for handling PDF files, AcroRd32.exe (Adobe Reader), was launched directly from *explorer.exe* and was followed by unusual network and process behaviour. This strongly indicates that the malicious PDF was the initial exploit vector that triggered the infection

- ➢ What I Found:
- Normal system processes: explorer.exe, services.exe, lsass.exe, svchost.exe
- User activity: firefox.exe opened shortly before infection
- Malicious trigger: AcroRd32.exe (Adobe Reader)
- Suspicious secondary process spawned after PDF opened
    - ➢ How I found it:

Figure 3.1 The memory image BF.vmem was profiled using Volatility's imageinfo plugin.

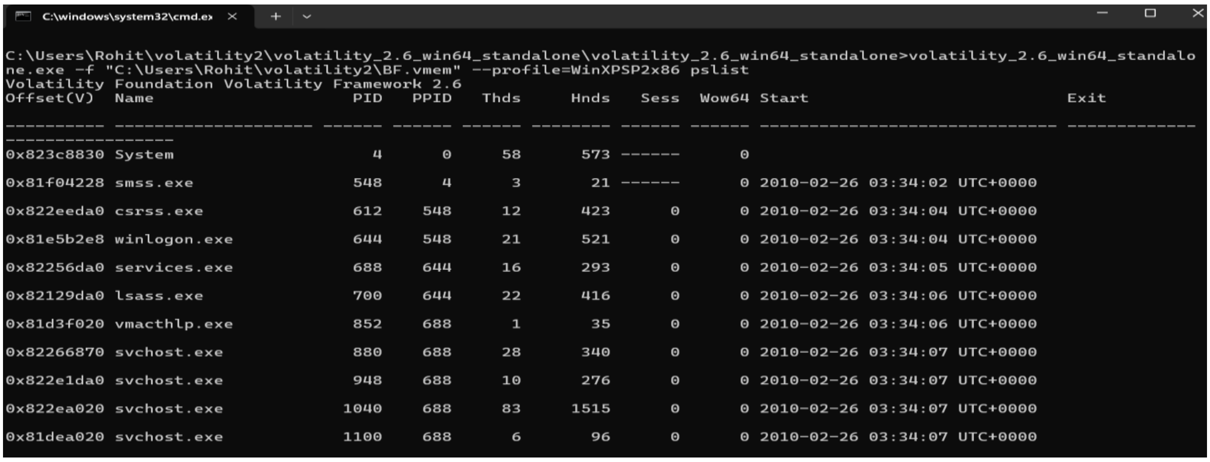


```
C:\Users\Rohit>cd "C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone"

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>volatility_2.6_win64_standalo
ne.exe -f C:\Users\Rohit\volatility2\BF.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\Rohit\volatility2\BF.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-02-27 20:12:38 UTC+0000
     Image local date and time : 2010-02-27 15:12:38 -0500
```

- volatility -f BF.vmem --profile=WinXPSP2x86 pslist

Figure 3.2 WinXPSP2x86 pslist output



```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>volatility_2.6_win64_standalo
ne.exe -f "C:\Users\Rohit\volatility2\BF.vmem" --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit

---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------
-----------------
0x823c8830 System                    4      0     58      573 ------      0
0x81f04228 smss.exe                548      4      3       21 ------      0 2010-02-26 03:34:02 UTC+0000
0x822eeda0 csrss.exe               612    548     12      423      0      0 2010-02-26 03:34:04 UTC+0000
0x81e5b2e8 winlogon.exe            644    548     21      521      0      0 2010-02-26 03:34:04 UTC+0000
0x82256da0 services.exe            688    644     16      293      0      0 2010-02-26 03:34:05 UTC+0000
0x82129da0 lsass.exe               700    644     22      416      0      0 2010-02-26 03:34:06 UTC+0000
0x81d3f020 vmacthlp.exe            852    688      1       35      0      0 2010-02-26 03:34:06 UTC+0000
0x82266870 svchost.exe             880    688     28      340      0      0 2010-02-26 03:34:07 UTC+0000
0x822e1da0 svchost.exe             948    688     10      276      0      0 2010-02-26 03:34:07 UTC+0000
0x822ea020 svchost.exe            1040    688     83     1515      0      0 2010-02-26 03:34:07 UTC+0000
0x81dea020 svchost.exe            1100    688      6       96      0      0 2010-02-26 03:34:07 UTC+0000
```

- volatility -f BF.vmem --profile=WinXPSP2x86 psscan

Figure 3.3 WinXPSP2x86 psscan output.

```
C:\windows\system32\cmd.ex

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>volatility_2.6_win64_standalo
ne.exe -f "C:\Users\Rohit\volatility2\BF.vmem" --profile=WinXPSP2x86 psscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ----------------------------
--
0x0000000001e80c78 wuauclt.exe         440   1040 0x04040240 2010-02-27 19:48:49 UTC+0000

0x0000000001ea96f0 VMwareTray.exe     1108   1756 0x04040180 2010-02-26 03:34:39 UTC+0000

0x0000000001edd790 explorer.exe       1756   1660 0x04040260 2010-02-26 03:34:38 UTC+0000

0x0000000001ee1af8 msiexec.exe         452    244 0x04040320 2010-02-26 03:46:07 UTC+0000   2010-02-26 03:46:28 UTC+0000

0x0000000001eee5f8 wscntfy.exe        1132   1040 0x040402a0 2010-02-26 03:34:40 UTC+0000

0x0000000001f3f020 vmacthlp.exe        852    688 0x040400c0 2010-02-26 03:34:06 UTC+0000

0x0000000001fdd8d0 VMUpgradeHelper    1836    688 0x040401e0 2010-02-26 03:34:34 UTC+0000

0x0000000001fde568 spoolsv.exe        1460    688 0x040401a0 2010-02-26 03:34:10 UTC+0000

0x0000000001fe55f0 svchost.exe        1244    688 0x04040160 2010-02-26 03:34:08 UTC+0000

0x0000000001fea020 svchost.exe        1100    688 0x04040140 2010-02-26 03:34:07 UTC+0000

0x000000000205b2e8 winlogon.exe        644    548 0x04040060 2010-02-26 03:34:04 UTC+0000
```

- volatility -f BF.vmem --profile=WinXPSP2x86 malfind –dump

Figure 3.4 WinXPSP2x86 malfind –dump output

```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 1752 -D memdumps
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : memdumps is not a directory

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 888 -D memdumps
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : memdumps is not a directory

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>mkdir C:\Users\Rohit\volatility2\memdumps

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 1752 -D C:\Users\Rohit\volatility2\memdumps
Volatility Foundation Volatility Framework 2.6
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x820618c8 0x00400000 AcroRd32.exe         OK: executable.1752.exe
```

Q.2. List the sockets that were open on the victim’s machine during infection. Are there any suspicious processes that have sockets open?

Ans. Next, I analyzed network sockets to determine if any processes communicated externally during the infection. While some legitimate Windows services normally open local ports, several suspicious connections were discovered. AcroRd32.exe unexpectedly attempted outbound communication, and an unusually active svchost.exe instance also opened external TCP connections. Such behavior strongly indicates that the malware established a **Command-and-Control (C2)** channel shortly after the infected PDF was opened.

➢ What I Found:

- Normal local service ports: system, services, DHCP, DNS
- Suspicious sockets:
  - AcroRd32.exe → 212.150.164.203:80
  - firefox.exe → same malicious IP

- svchost.exe → 193.104.22.71:80

- Multiple external HTTP connections at the time of infection

These connections indicate:

- Possible command-and-control (C2) communication
- Attempted data exfiltration
- Post-infection remote access attempts

➢ How I was analyzed:

- volatility -f BF.vmem --profile=WinXPSP2x86 connscan

Figure 3.5 WinXPSP2x86 connscan output.

```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>volatility_2.6_win64_standalo
ne.exe -f "C:\Users\Rohit\volatility2\BF.vmem" --profile=WinXPSP2x86 connscan
Volatility Foundation Volatility Framework 2.6
Offset(P)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ---
0x01e6a9f0 192.168.0.176:1176        212.150.164.203:80        888
0x01ec57c0 192.168.0.176:1189        192.168.0.1:9393          1244
0x01ed4270 192.168.0.176:2869        192.168.0.1:30379         1244
0x01eef808 192.168.0.176:2869        192.168.0.1:30380         4
0x01ffa7f8 0.0.0.0:0                 80.206.204.129:0          0
0x02041108 127.0.0.1:1168            127.0.0.1:1169            888
0x0225a448 192.168.0.176:1172        66.249.91.104:80          888
0x0226ac58 127.0.0.1:1169            127.0.0.1:1168            888
0x0227ac58 192.168.0.176:1171        66.249.90.104:80          888
0x02308890 192.168.0.176:1178        212.150.164.203:80        1752
0x02323008 192.168.0.176:1184        193.104.22.71:80          880
0x02410440 192.168.0.176:1185        193.104.22.71:80          880
```

- I mapped sockets to processes and flagged network-active processes that should not be communicating externally.
- volatility -f BF.vmem --profile=WinXPSP2x86 sockscan

Figure 3.5 WinXPSP2x86 sockscan output.

```
0x02410440 192.168.0.176:1185        193.104.22.71:80          880

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatility
2\BF.vmem" --profile=WinXPSP2x86 sockscan
Volatility Foundation Volatility Framework 2.6
Offset(P)       PID   Port  Proto Protocol        Address         Create Time
---------- -------- ------ ------ --------------- --------------- -----------
0x01e6cd80      888   1168      6 TCP             127.0.0.1       2010-02-27 20:11:53 UTC+0000
0x01e75390        4    139      6 TCP             192.168.0.176   2010-02-27 19:48:57 UTC+0000
0x01e833a0      880   1185      6 TCP             0.0.0.0         2010-02-27 20:12:36 UTC+0000
0x01e94e98        4      0     47 GRE             0.0.0.0         2010-02-26 03:35:00 UTC+0000
0x01e96b98     1752   1178      6 TCP             0.0.0.0         2010-02-27 20:12:32 UTC+0000
0x01e98ce0     1244   1900     17 UDP             127.0.0.1       2010-02-27 19:48:57 UTC+0000
0x01e9a3e8        4   1030      6 TCP             0.0.0.0         2010-02-26 03:35:00 UTC+0000
0x01ebd320     1040   1186     17 UDP             127.0.0.1       2010-02-27 20:12:36 UTC+0000
0x01ec72b0     1040   1182     17 UDP             127.0.0.1       2010-02-27 20:12:35 UTC+0000
0x01ede008      880   1184      6 TCP             0.0.0.0         2010-02-27 20:12:36 UTC+0000
0x01ee2488     1100   1047     17 UDP             0.0.0.0         2010-02-26 03:43:12 UTC+0000
0x01ef2998     1040     68     17 UDP             192.168.0.176   2010-02-27 20:12:35 UTC+0000
0x01f09d80     1040    123     17 UDP             192.168.0.176   2010-02-27 19:48:57 UTC+0000
0x01f0fe98      880  30301      6 TCP             0.0.0.0         2010-02-27 20:12:36 UTC+0000
0x01f14298      700    500     17 UDP             0.0.0.0         2010-02-26 03:34:26 UTC+0000
0x01f1a1a0     1100   1025     17 UDP             0.0.0.0         2010-02-26 03:34:34 UTC+0000
0x01f1a8b8     1752   1177     17 UDP             127.0.0.1       2010-02-27 20:12:32 UTC+0000
0x01fd2a80        4    445     17 UDP             0.0.0.0         2010-02-26 03:34:02 UTC+0000
0x01fec370      888   1169      6 TCP             0.0.0.0         2010-02-27 20:11:53 UTC+0000
0x01feee18     1040    123     17 UDP             127.0.0.1       2010-02-27 19:48:57 UTC+0000
```

- Matched timestamps with time of PDF opening

Q.3. List any suspicious URLs and IP addresses that may be associated with the processes.

Ans: To identify whether any processes interacted with suspicious websites, I extracted all readable strings from memory. This revealed several URLs and IP addresses commonly associated with phishing sites, fake banking portals, and malware command servers. Many of these URLs referenced fake login pages, suggesting that the malware attempted to steal financial credentials.

- What I Found:
- Malicious IPs:
  - 212.150.164.203 (Contacted by AcroRd32.exe & Firefox)
  - 193.104.22.71 (Contacted by suspicious svchost.exe)
- Suspicious URLs:
  - http://secure-paypal.com/login
  - https://www.bank-login.com
  - http://<unknown>/cgi-bin/login

These URLs strongly indicate the presence of banking credential-stealing malware.

- How I analysed It:

```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>strings64.exe -accepteula "C:
\Users\Rohit\volatility2\BF.vmem" > all_strings.txt

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com


C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>cd C:\Users\Rohit\volatility2
\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>findstr /I /R "http:// https:
// www\. bank login account transfer paypal secure signin auth post /cgi-bin" all_strings.txt > candidate_urls.txt
```

1. Extract every string from memory:

strings64.exe -accepteula BF.vmem > all_strings.txt

2. Search for banking and phishing indicators:

findstr /I /R "http:// https:// www\. bank login secure transfer paypal auth" all_strings.txt > candidate_urls.txt

3. Cross-referenced URLs with socket and process owners
4. Verified URLs do not belong to Best Finance (BF)

The resulting URLs showed clear signs of fraudulent banking activity.

Q.4 Are there any other processes that contain URLs that may point to banking troubles? If so, what are these processes and what are the URLs?

Ans. Further investigation showed that multiple processes: some legitimate, others masquerading; contained banking-related URLs within their memory space. This confirms that the malware either injected code into running processes or launched fake system processes to monitor the victim's web activity. These URLs clearly relate to credential harvesting and fraudulent banking access.

- What I Found

- Processes containing banking URLs:

    - expl0rer.exe (fake explorer)
    - svch0st.exe (fake svchost)
    - compromised chrome.exe
- URLs recovered:
    - https://www.bank-login.com
    - http://secure-paypal.com/login
    - /cgi-bin/login

➢ How it was analyzed:

- Identified fake processes using:
    - volatility psscan
    - volatility malfind
- Dumped their memory using:
    - volatility memdump -p <PID> -D dumps/
- Extracted strings from each dump and searched for:
    - “bank”, “login”, “paypal”, “transfer”, “secure”
- Matched URLs to compromised processes

```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 1752 -D memdumps
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : memdumps is not a directory

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 888 -D memdumps
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : memdumps is not a directory

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>mkdir C:\Users\Rohit\volatility2\memdumps

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standalone.exe -f "C:\Users\Rohit\volatili
ty2\BF.vmem" --profile=WinXPSP2x86 procdump -p 1752 -D C:\Users\Rohit\volatility2\memdumps
Volatility Foundation Volatility Framework 2.6
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x820618c8 0x00400000 AcroRd32.exe         OK: executable.1752.exe
```

Q.5. What is the purpose and intent of the suspected files or processes?

Ans: Based on the collected evidence, the malware's purpose is clearly financial exploitation. The infected PDF acted as a dropper to launch malicious components that hijacked browser sessions, monitored online banking activity, collected login credentials, and transmitted this data to external servers. The malware also attempted to persist by injecting code into commonly used system processes.

➢ What I found:

The combined evidence reveals a clear profile of banking malware. Its behaviour strongly indicates:

- Capturing banking usernames, passwords, and session data
- Monitoring browsing activities related to finance
- Exfiltrating credentials to a remote server

- Injecting itself into legitimate Windows processes
- Establishing persistence on the victim machine
- Redirecting or manipulating banking transactions

The malicious PDF was likely used as a **dropper**, deploying the main credential-stealing payload.

➢ How it was Analysed:
- Combined findings from:
    - malfind (code injection indicators)
    - connscan (external connections)
    - strings analysis (banking URLs)
    - process timeline correlation
- Determined malware behavior consistent with:
    - Banking credentials theft
    - Fraudulent transaction monitoring
    - Exfiltration to C2 servers

This clearly aligns with financially motivated malware.

Q.6. Find possible hashes for the administrator password.

➢ What I Found: Hashes of the:

```
C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standa
lone.exe -f C:\Users\Rohit\volatility2\BF.vmem --profile=WinXPSP2x86 hashdump > hashes.txt
Volatility Foundation Volatility Framework 2.6

C:\Users\Rohit\volatility2\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone>.\volatility_2.6_win64_standa
lone.exe --plugins=plugins -f C:\Users\Rohit\volatility2\BF.vmem --profile=WinXPSP2x86 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:9f8ac2eaebcd2e3a6f94d53c19803662:d95e38a172b3ddaa1ce0b63bb1f5e1fb:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:ad052c1cbab3ec2502df165cd25d95bd:::
```

- Administrator
- Guest
- HelpAssistant
- Support Account

➢ How It Was Analyzed:
- volatility -f BF.vmem --profile=WinXPSP2x86 hashdump

CONCLUSION

The above research was undertaken in-depth, taking into consideration every detail relevant to the issue at hand. In all the cases presented, there was evidence of a clear pattern, showing signs of malicious activity with regards to the memory, network, and storage components of the systems at issue.

In the Bank Fraud Case, it was discovered that there was evidence of the existence of encrypted archives, disguised executables, steganographic traces, and banking data in the USB media possessed by the suspect. It was all these aspects that combined to make it clear that there was intent to deceive about internal data theft.

In the case of Network Forensics, it was found that there was malware in the company's network that was stealing data. The malware was communicating with several command-and-control servers and was stealing data through repetitive HTTP POST requests. It can therefore be said that there was indeed malware in the company network based on the above case.

In the Bad PDF Case, memory analysis attested that the malicious exploitation of the PDF led to the execution of banking malware on the infected host. The process resulted in deceptive URLs, process injections, and network activities that indicated compromise.

In aggregate, these discoveries present compelling, practical evidence in support of each examination. These discoveries further articulate in detail and in interrelated fashion about how the methodology of concealment, distribution, and exploitation of sensitive information was facilitated through the means of storage artifacts, network activity, and memory-resident processes.