***Laboratory 3: Artifact Acquisition and Parser***

Group 17 – Malia  
2024-06-19

Table of Contents

[Change Control iii](#_Toc169730658)

[I. Objectives 1](#_Toc169730659)

[II. Activity Screenshots 2](#_Toc169730660)

[A. Acquiring Memory Dump using KAPE and Magnet RAM Capture 2](#_Toc169730661)

[B. Acquiring Data Triage (One-by-one using Python) 4](#_Toc169730662)

[III. Conclusion 10](#_Toc169730663)

[IV. Appendix 11](#_Toc169730664)

[A. REG\_AC\_TOOL.py (Registry Acquisition Tool) 11](#_Toc169730665)

[B. REG\_PARSE\_TOOL.py (Registry Parser Tool) 12](#_Toc169730666)

[V. References 14](#_Toc169730667)

[Figure 1. Successful memory dump capture of Win11 Machine via KAPE 2](#_Toc169730668)

[Figure 2. Console Logs of KAPE Command (in txt) 2](#_Toc169730669)

[Figure 3. Win11 memory dump captured via KAPE 3](#_Toc169730670)

[Figure 4. md5 and Volatility comparison of Win11 machine memory dump 3](#_Toc169730671)

[Figure 5. Registry Acquisition Tool successfully turned into exe file 4](#_Toc169730672)

[Figure 6. Registry Acquisition Tool successfully executed in Victim Machine via RDP (default, sam, security, software and system hives were collected) 4](#_Toc169730673)

[Figure 7. DEFAULT, SAM, SECURITY, SOFTWARE and SYSTEM hives were collected via FTK Imager in Victim Machine via RDP 5](#_Toc169730674)

[Figure 8. Hives collected via Registry Acquisition Tool were successfully parsed using Registry Parsing Tool 5](#_Toc169730675)

[Figure 9. Hives collected via FTK Imager were successfully parsed using Registry Parsing Tool 6](#_Toc169730676)

[Figure 10. Side-by-side comparison of hives collected and parsed via Registry Acquisition Tool (on the left) and FTK Imager (on the right) 6](#_Toc169730677)

[Figure 11. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “SAM\Domains\Account\Users”) 7](#_Toc169730678)

[Figure 12. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “Control Panel\Accessibility”) 7](#_Toc169730679)

[Figure 13. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “Policy\Accounts”) 8](#_Toc169730680)

[Figure 14. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “SAM\Domains\Account\Users”) 8](#_Toc169730681)

[Figure 15. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “Control Panel\Accessibility”) 9](#_Toc169730682)

[Figure 16. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “Policy\Accounts”) 9](#_Toc169730683)

# Change Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Updates | Author | Date |
| 0.1 | Draft | Gideon Andrew L. Malia | YYYY-MM-DD |
| 0.2 | Working Document | Gideon Andrew L. Malia | 2024-06-19 |
| 0.x |  |  |  |
| 1.0 |  |  |  |

# Objectives

In this laboratory activity, the group aims:

1. to successfully acquire a memory dump from a Windows 11 system using KAPE and Magnet RAM Capture;
2. to develop and execute a Python-based tool for acquiring and parsing Windows Registry hives, capable of operating in both live and offline environments; and
3. to validate the parsed registry data and compare it with results obtained from established tools like RECmd and Registry Explorer.

# Activity Screenshots

## Acquiring Memory Dump using KAPE and Magnet RAM Capture

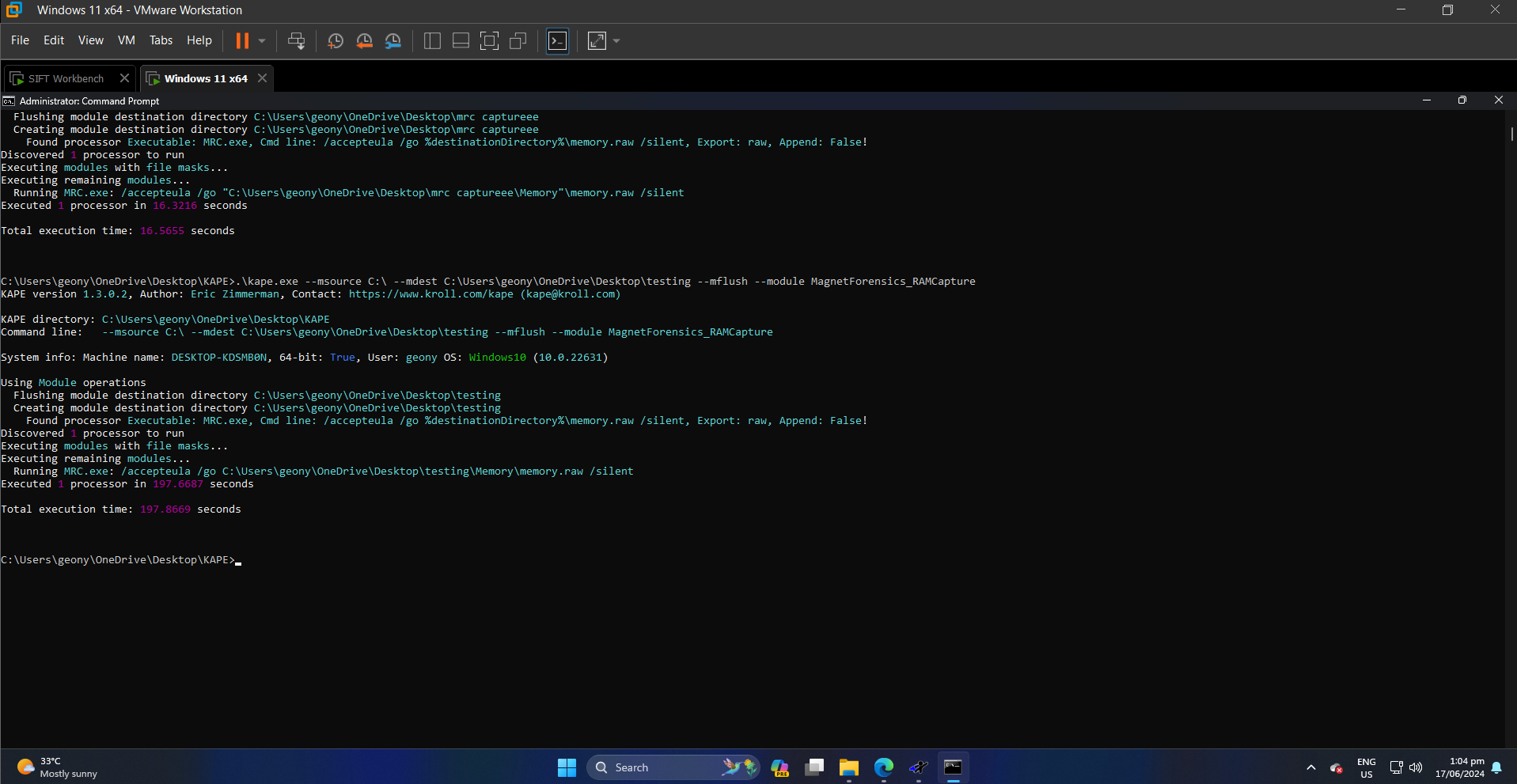


Figure 1. Successful memory dump capture of Win11 Machine via KAPE

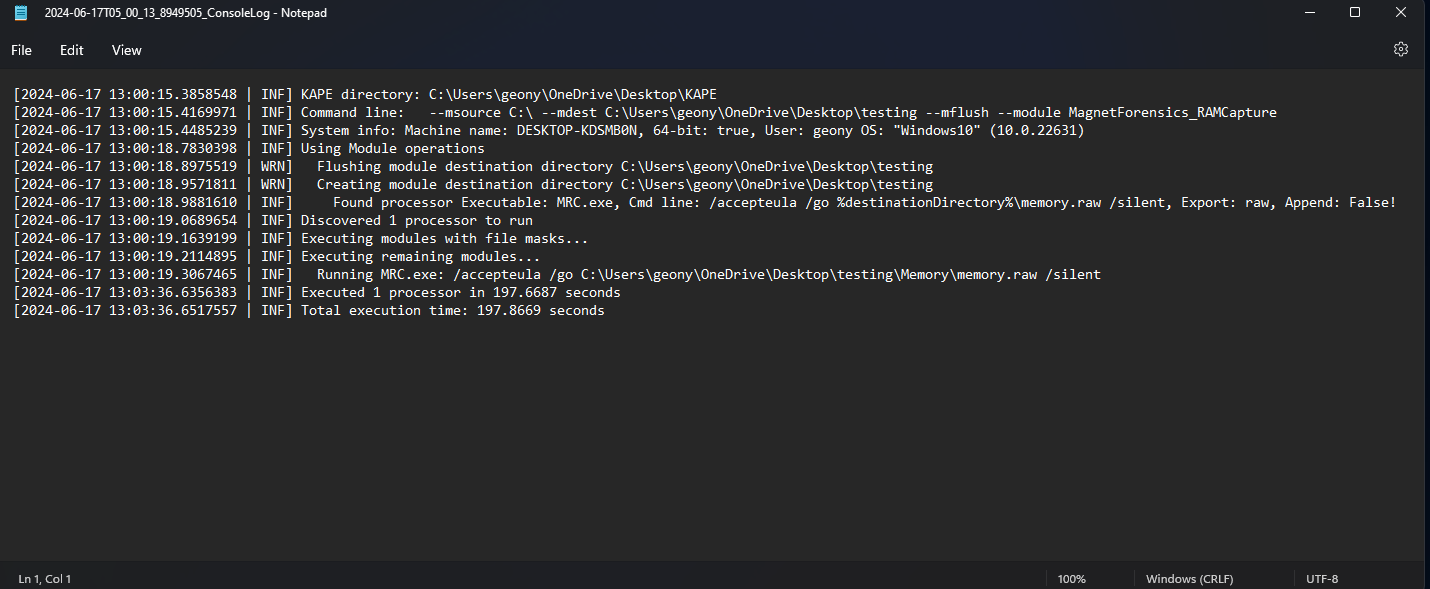


Figure 2. Console Logs of KAPE Command (in txt)

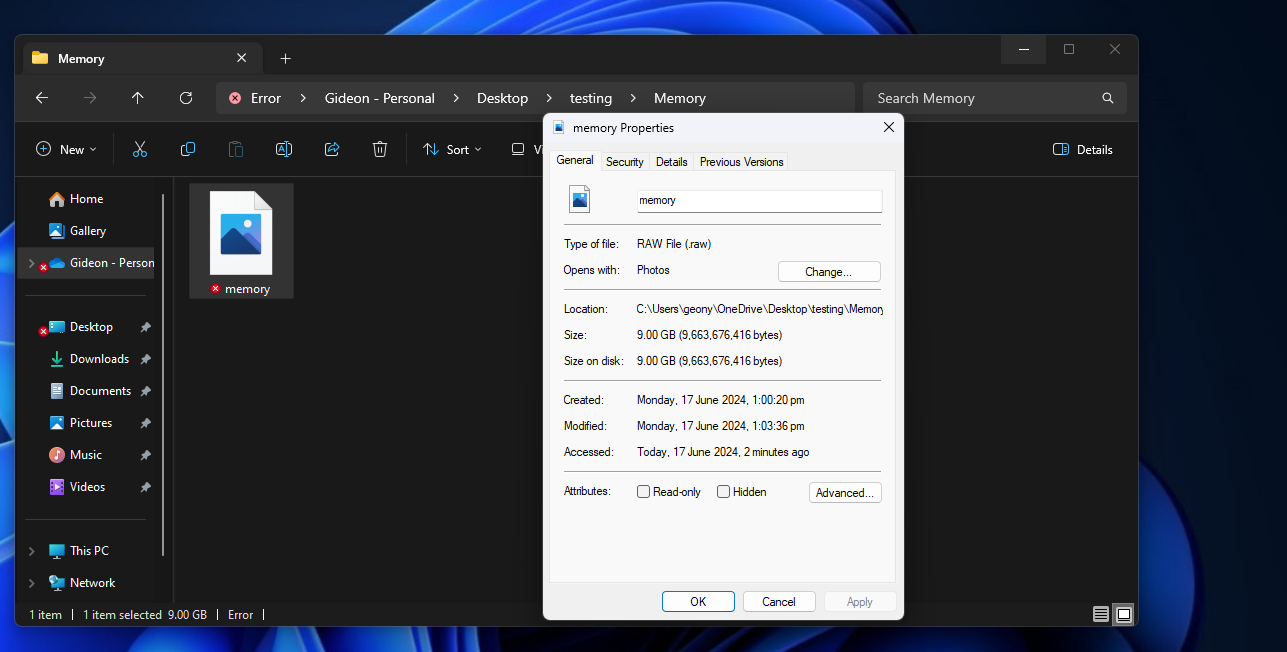


Figure 3. Win11 memory dump captured via KAPE

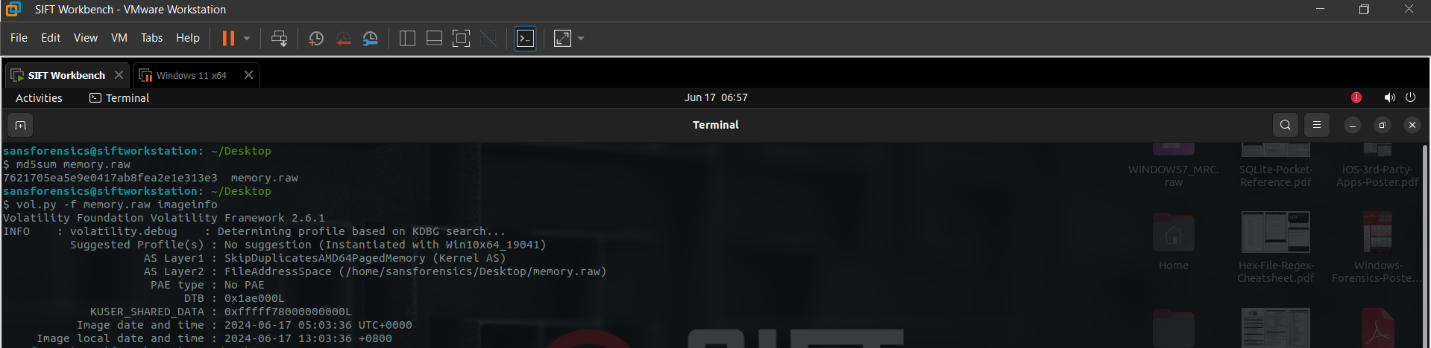


Figure 4. md5 and Volatility comparison of Win11 machine memory dump

## Acquiring Data Triage (One-by-one using Python)

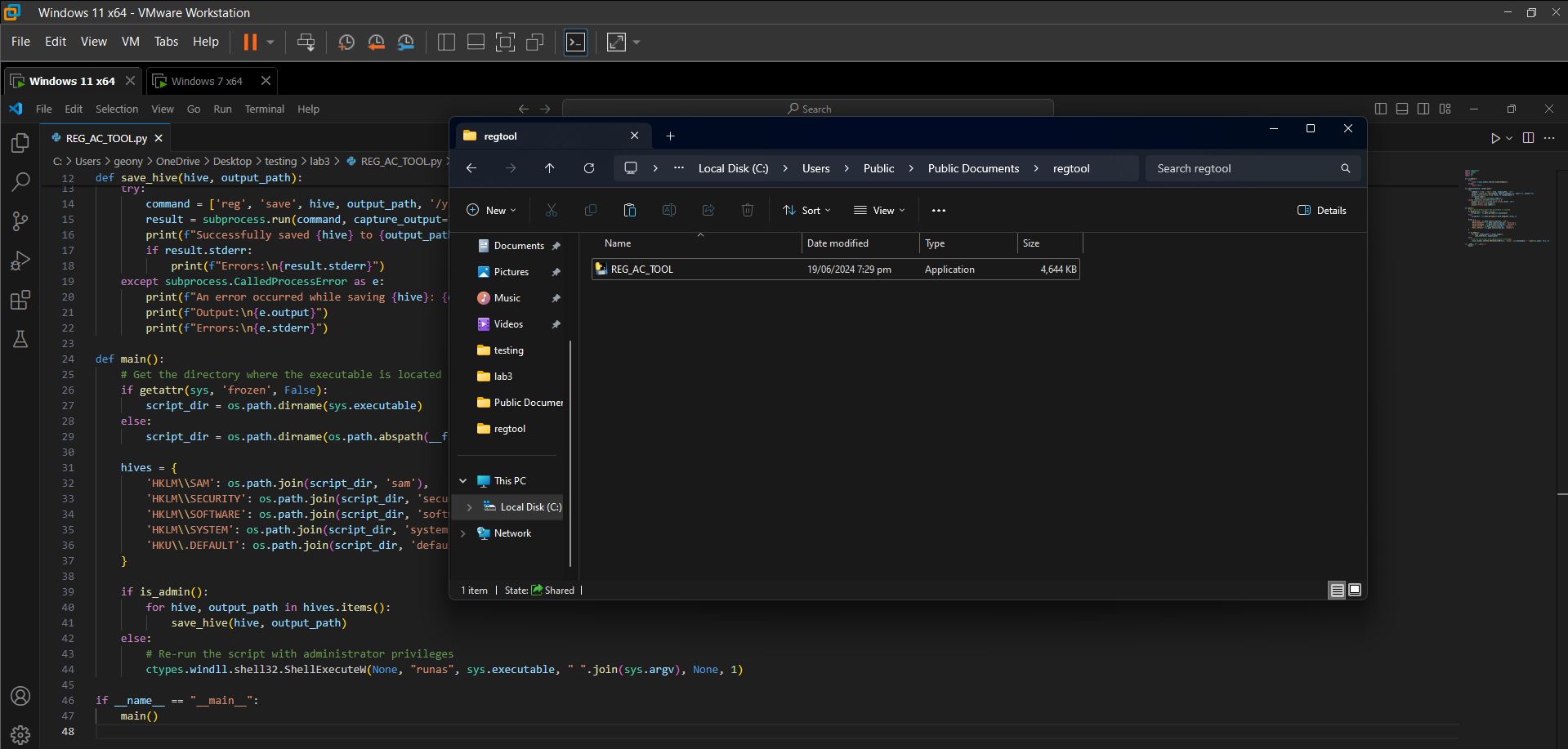


Figure 5. Registry Acquisition Tool successfully turned into exe file

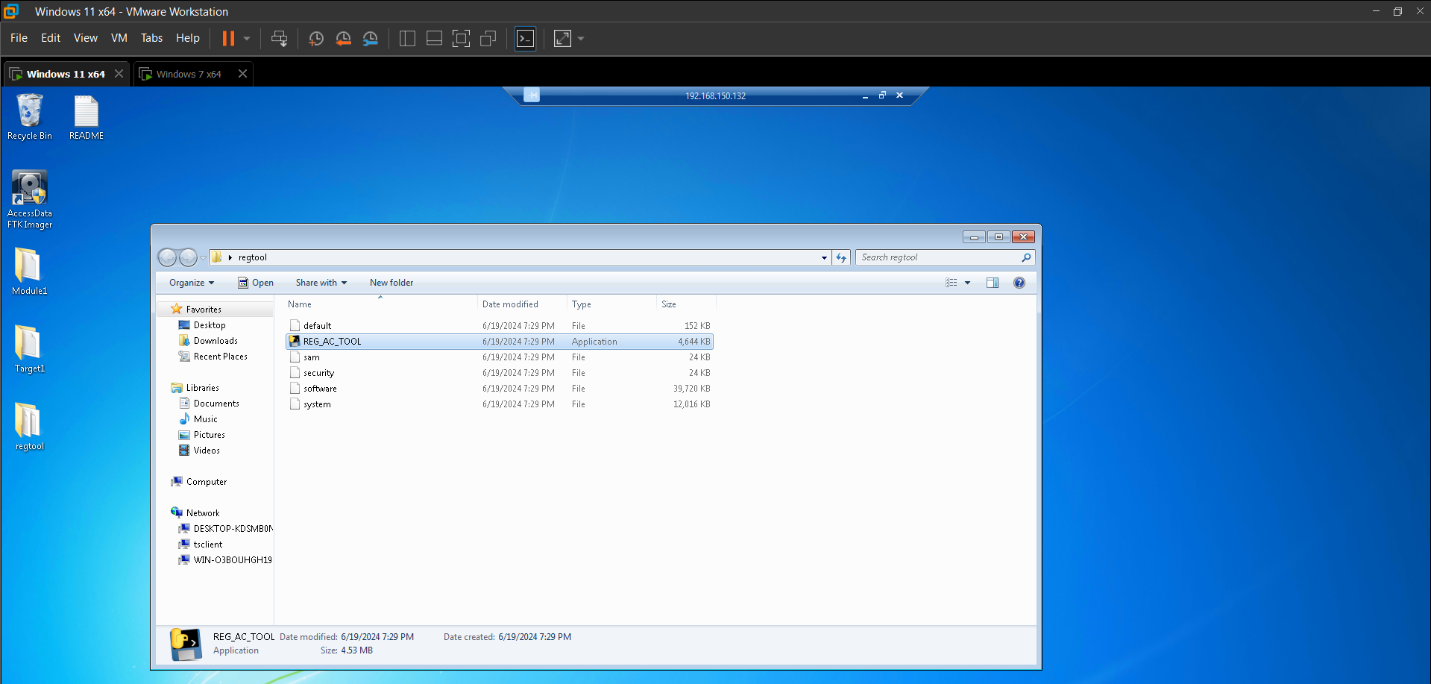


Figure 6. Registry Acquisition Tool successfully executed in Victim Machine via RDP (default, sam, security, software and system hives were collected)

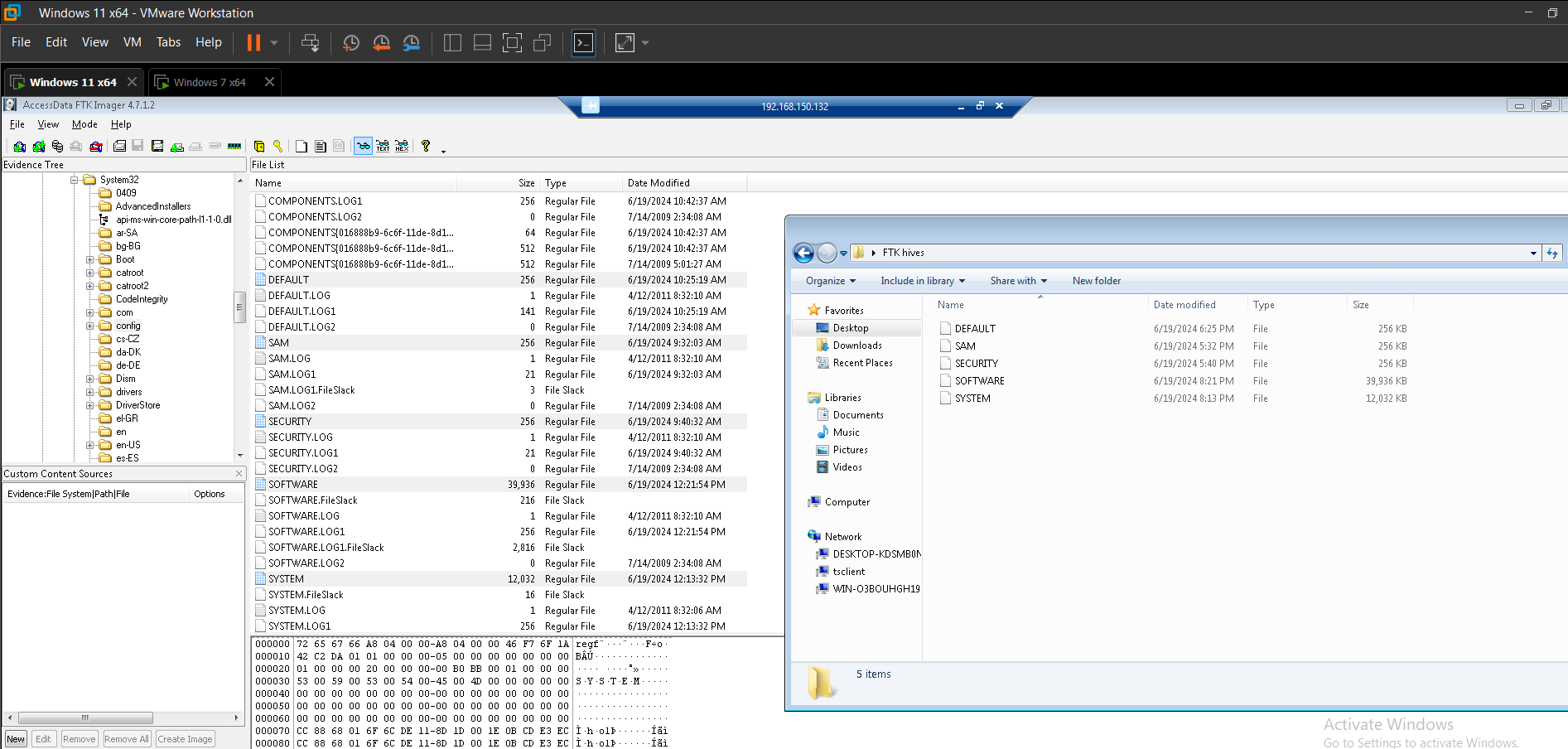


Figure 7. DEFAULT, SAM, SECURITY, SOFTWARE and SYSTEM hives were collected via FTK Imager in Victim Machine via RDP

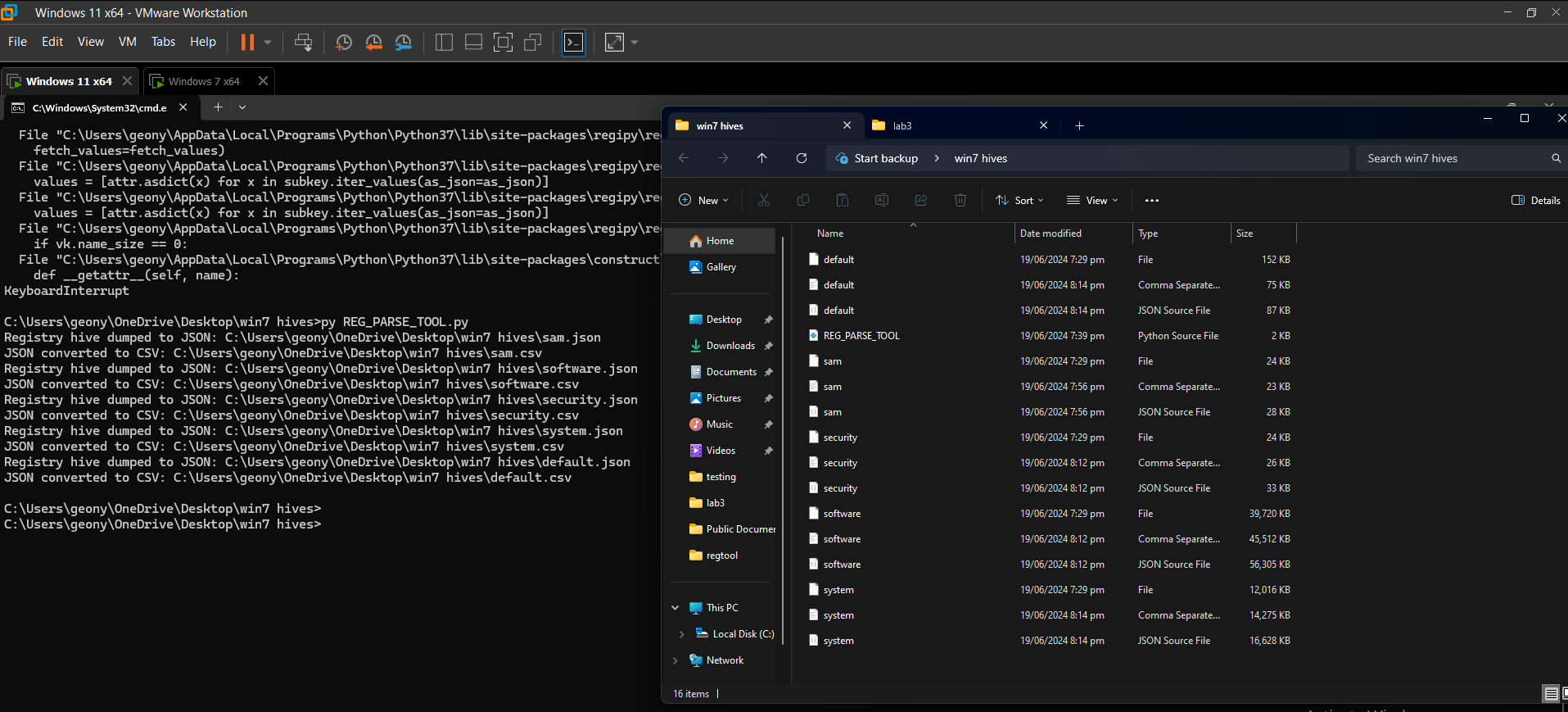


Figure 8. Hives collected via Registry Acquisition Tool were successfully parsed using Registry Parsing Tool

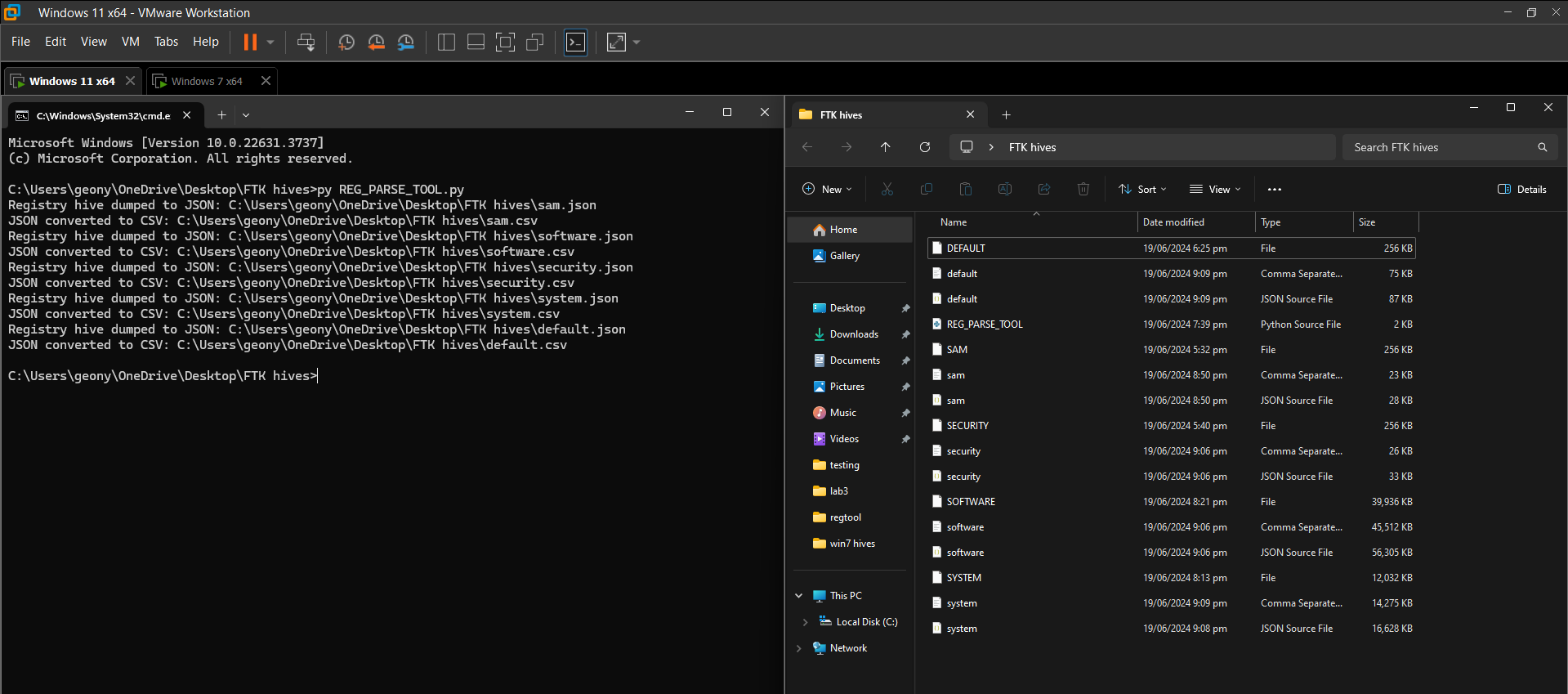


Figure 9. Hives collected via FTK Imager were successfully parsed using Registry Parsing Tool

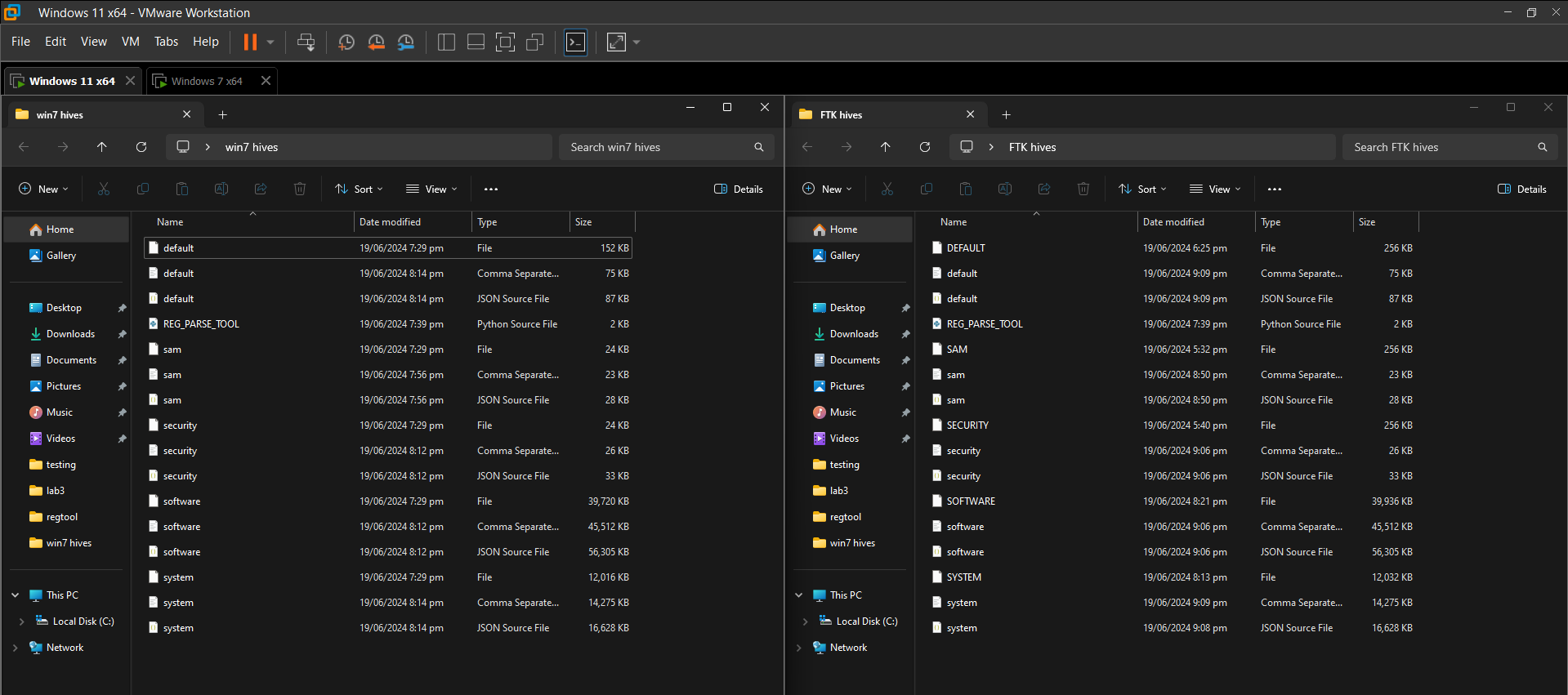


Figure 10. Side-by-side comparison of hives collected and parsed via Registry Acquisition Tool (on the left) and FTK Imager (on the right)

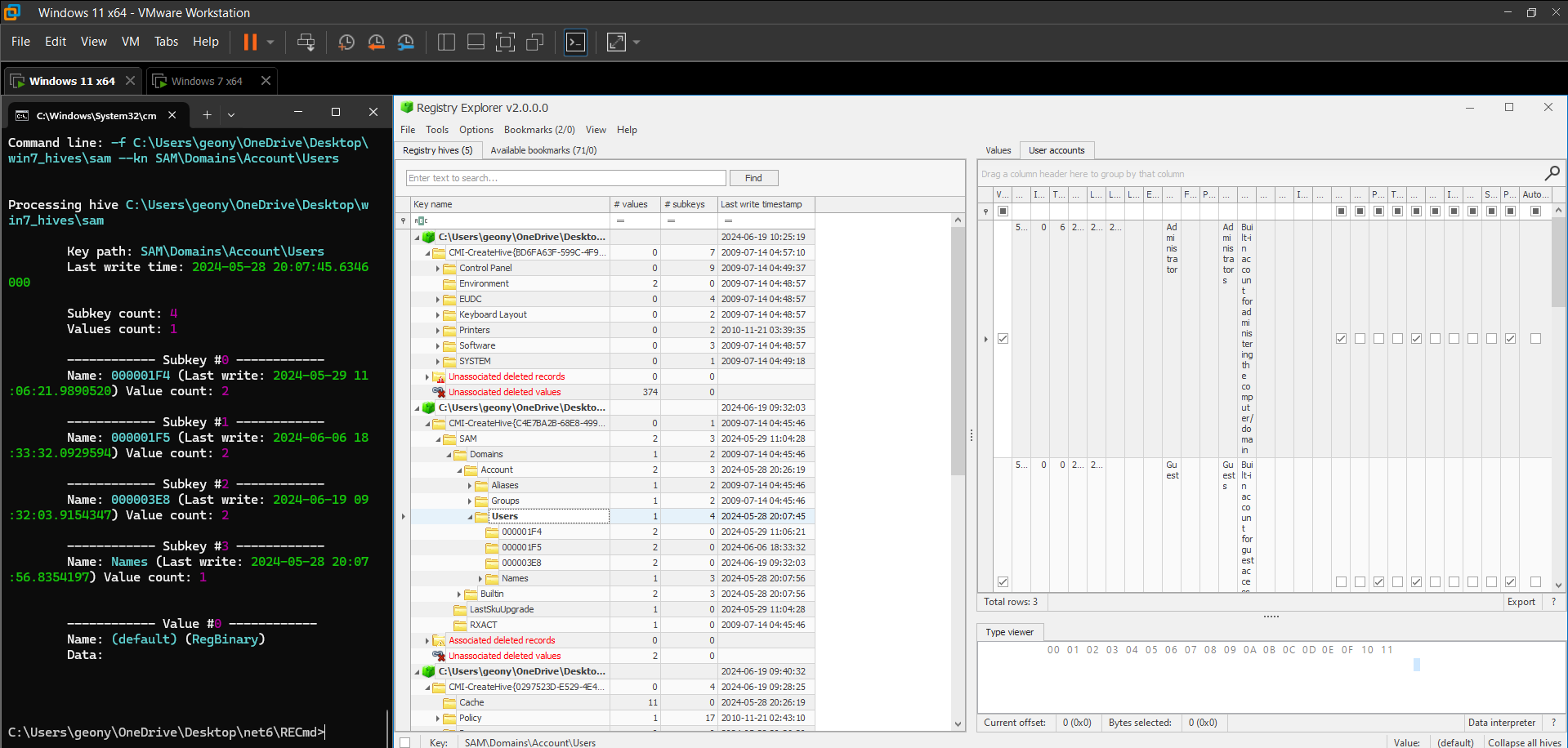


Figure 11. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “SAM\Domains\Account\Users”)

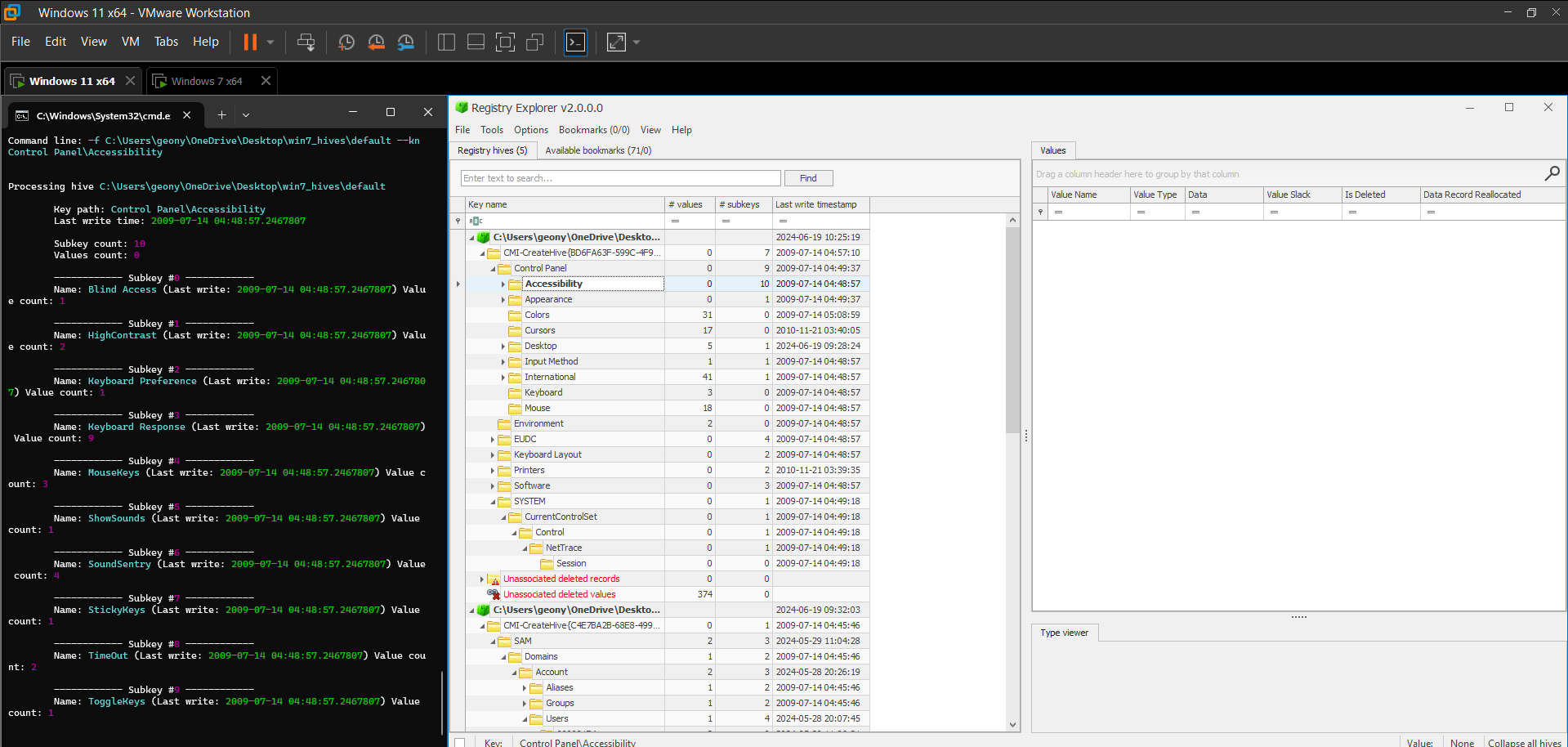


Figure 12. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “Control Panel\Accessibility”)

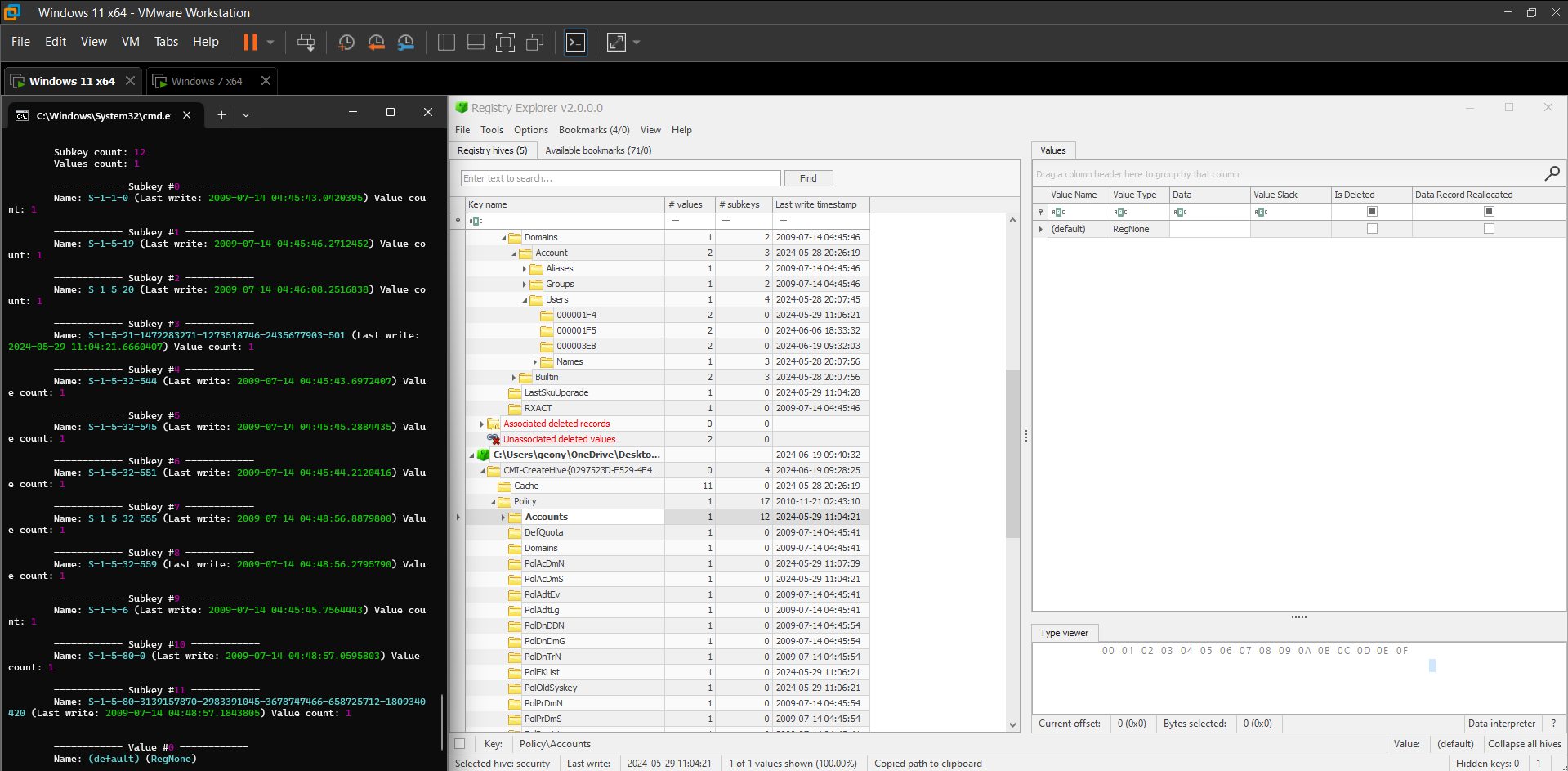


Figure 13. Side-by-side comparison of RECmd and Registry Explorer (hives via Registry Acquisition Tool; key: “Policy\Accounts”)

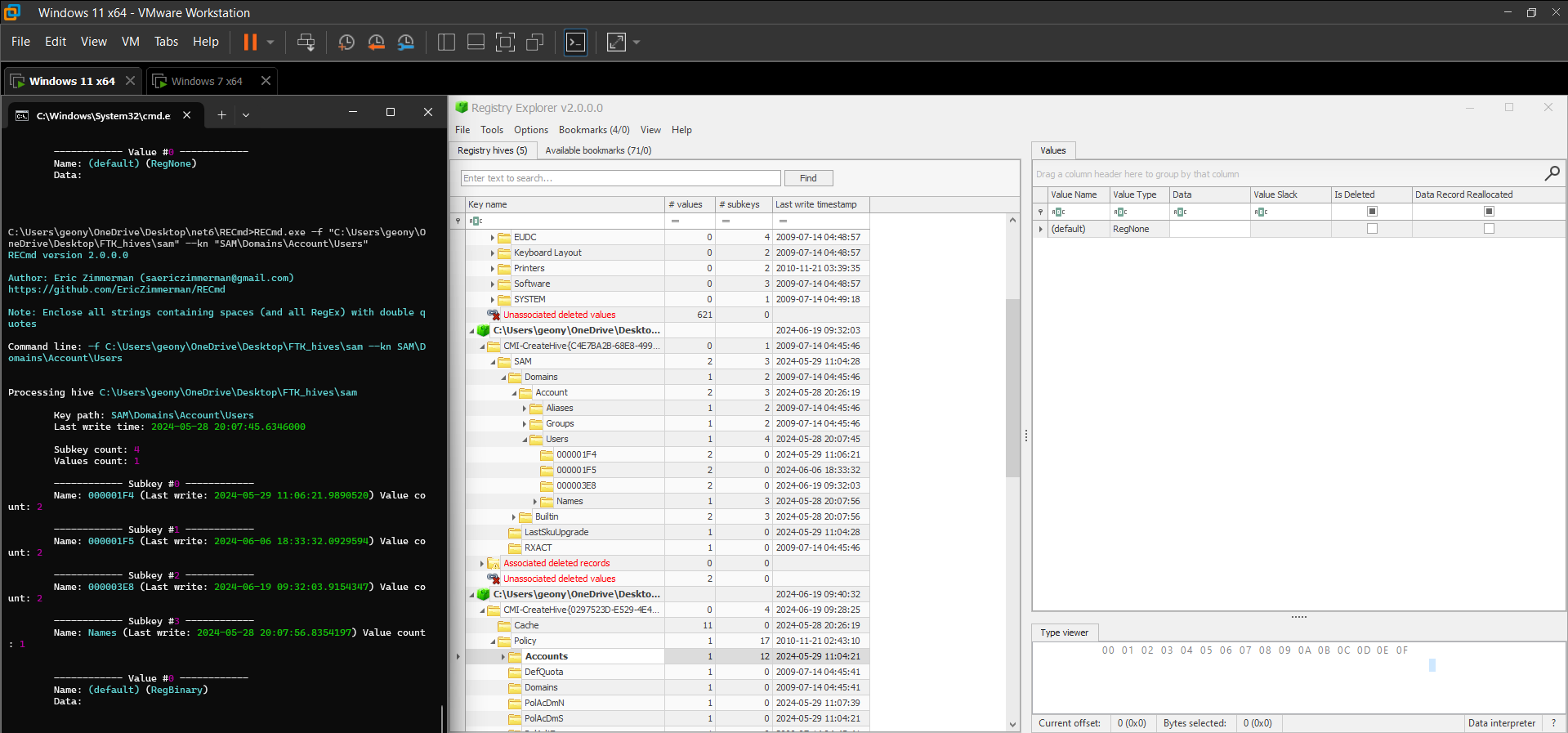


Figure 14. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “SAM\Domains\Account\Users”)

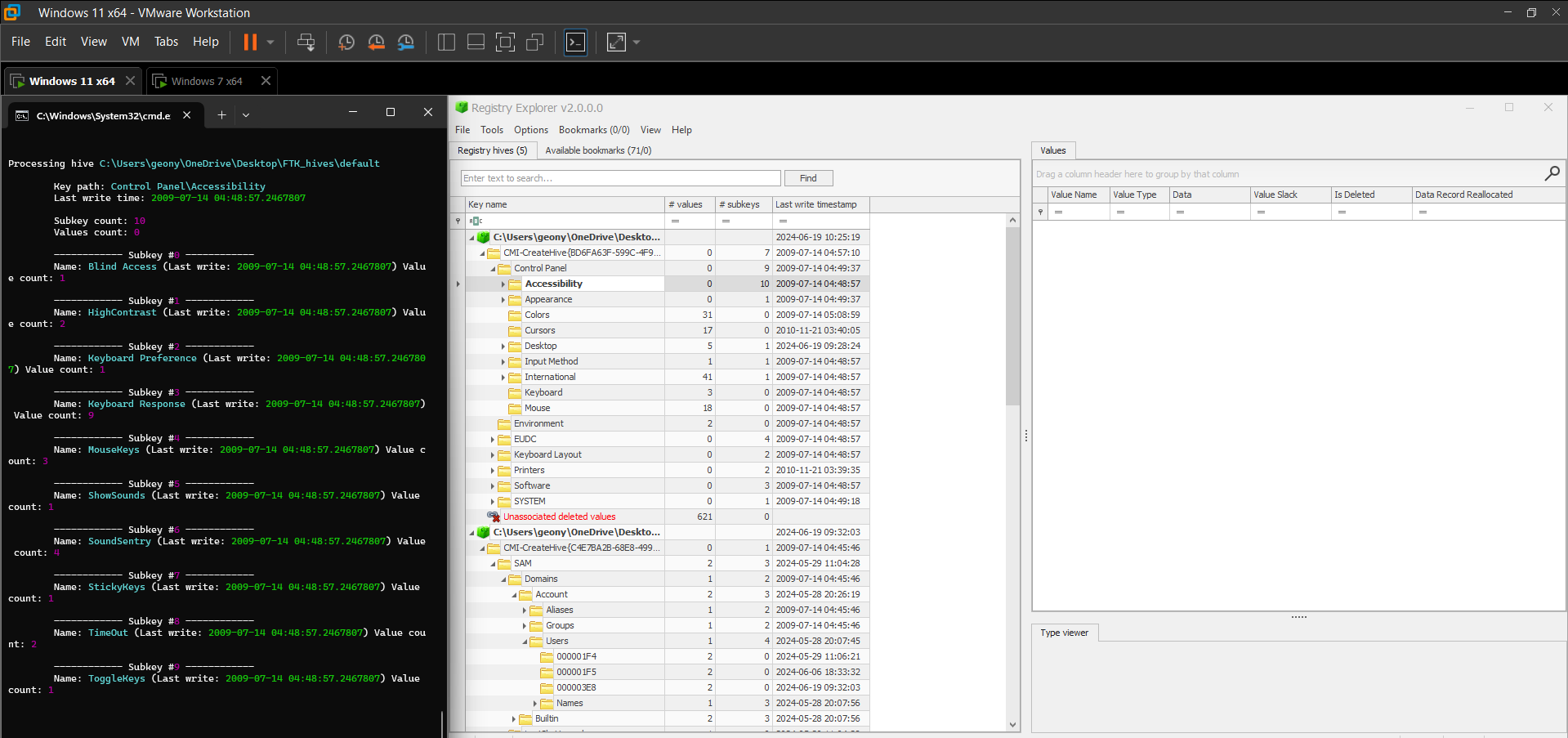


Figure 15. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “Control Panel\Accessibility”)

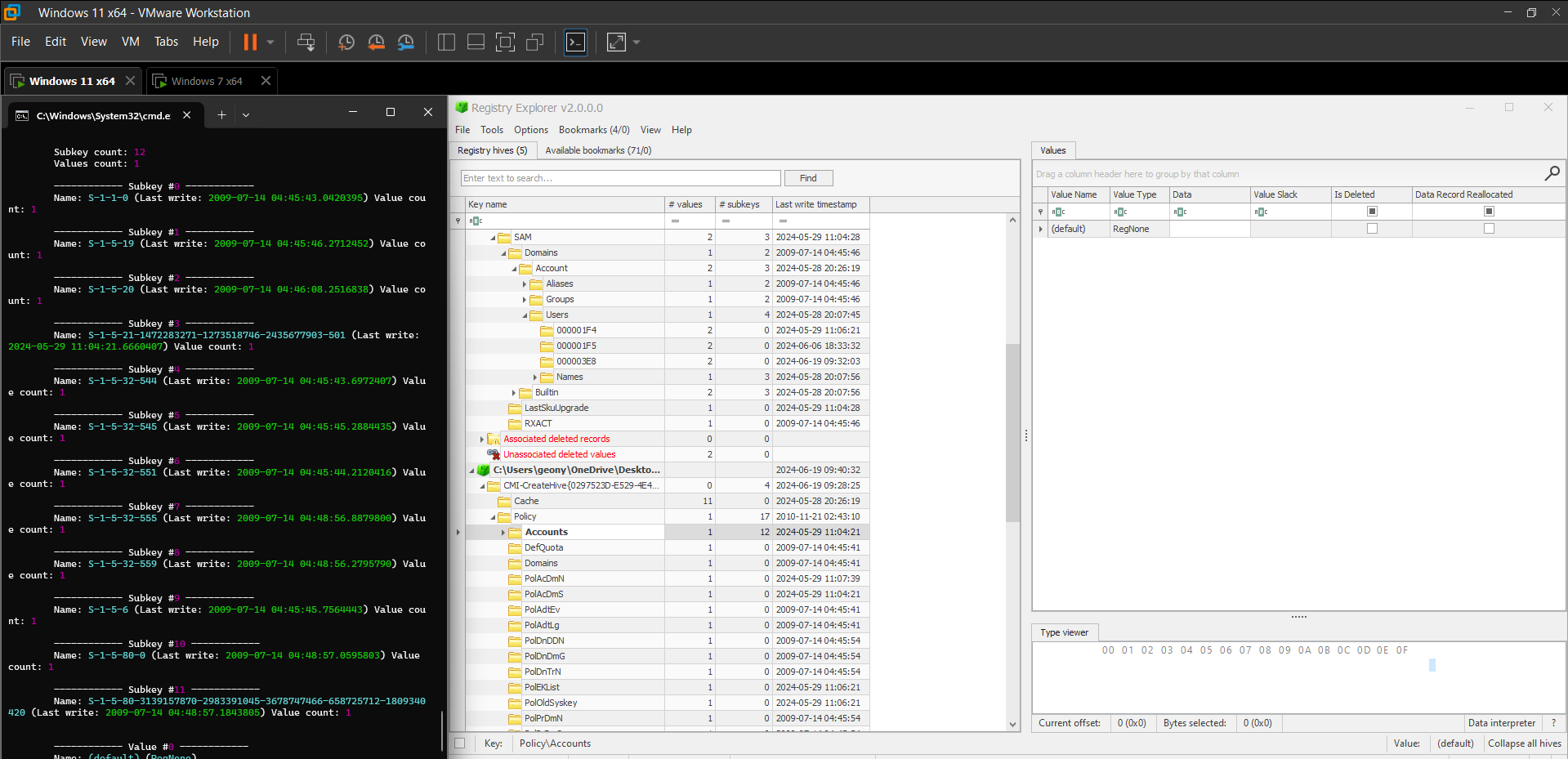


Figure 16. Side-by-side comparison of RECmd and Registry Explorer (hives via FTK Imager; key: “Policy\Accounts”)

From Figure 11 - Figure 16, the side-by-side comparison of RECmd and Registry explorer has shown similarities in terms of the number of values and subkeys of a certain key. There are no significant differences when collected via Registry Acquisition Tool or FTK Imager.

# Conclusion

The group successfully acquired a memory dump from the Windows 11 machine using KAPE and Magnet RAM Capture. The group also were able to code and execute a working Registry Acquisition Tool and Parser that can be executed in both Live and Offline setup. Since the output of the RECmd and Registry Explorer are similar when selecting a specific key, it has been proven that our created tool is more beneficial to use because it is more user-friendly and accessible than when using FTK Imager to collect registry hives.

# Appendix

## REG\_AC\_TOOL.py (Registry Acquisition Tool)

import subprocess

import ctypes

import sys

import os

def is\_admin():

    try:

        return ctypes.windll.shell32.IsUserAnAdmin()

    except:

        return False

def save\_hive(hive, output\_path):

    try:

        command = ['reg', 'save', hive, output\_path, '/y']

        result = subprocess.run(command, capture\_output=True, text=True, check=True)

        print(f"Successfully saved {hive} to {output\_path}")

        if result.stderr:

            print(f"Errors:\n{result.stderr}")

    except subprocess.CalledProcessError as e:

        print(f"An error occurred while saving {hive}: {e}")

        print(f"Output:\n{e.output}")

        print(f"Errors:\n{e.stderr}")

def main():

    # Get the directory where the executable is located

    if getattr(sys, 'frozen', False):

        script\_dir = os.path.dirname(sys.executable)

    else:

        script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

    hives = {

        'HKLM\\SAM': os.path.join(script\_dir, 'sam'),

        'HKLM\\SECURITY': os.path.join(script\_dir, 'security'),

        'HKLM\\SOFTWARE': os.path.join(script\_dir, 'software'),

        'HKLM\\SYSTEM': os.path.join(script\_dir, 'system'),

        'HKU\\.DEFAULT': os.path.join(script\_dir, 'default')

    }

    if is\_admin():

        for hive, output\_path in hives.items():

            save\_hive(hive, output\_path)

    else:

        # Re-run the script with administrator privileges

        ctypes.windll.shell32.ShellExecuteW(None, "runas", sys.executable, " ".join(sys.argv), None, 1)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

## REG\_PARSE\_TOOL.py (Registry Parser Tool)

import os

import pandas as pd

import json

from regipy.registry import RegistryHive

from regipy.plugins.utils import dump\_hive\_to\_json

def parse\_hives\_to\_json\_and\_csv(hive\_paths, output\_dir):

    for hive\_path in hive\_paths:

        # Load the registry hive

        hive = RegistryHive(hive\_path)

        # Create JSON output file

        json\_file = os.path.join(output\_dir, os.path.basename(hive\_path) + '.json')

        dump\_hive\_to\_json(hive, json\_file, name\_key\_entry=None)

        print(f"Registry hive dumped to JSON: {json\_file}")

        # Create CSV output file

        csv\_file = os.path.join(output\_dir, os.path.basename(hive\_path) + '.csv')

        with open(json\_file, 'r') as file:

            data\_list = [json.loads(line.strip()) for line in file if line.strip()]

        df = pd.DataFrame(data\_list)

        df.to\_csv(csv\_file, index=False)

        print(f"JSON converted to CSV: {csv\_file}")

def main():

    # Get current script directory

    current\_directory = os.path.dirname(os.path.realpath(\_\_file\_\_))

    # Specify hive file paths

    hive\_paths = [

        os.path.join(current\_directory, 'sam'),

        os.path.join(current\_directory, 'software'),

        os.path.join(current\_directory, 'security'),

        os.path.join(current\_directory, 'system'),

        os.path.join(current\_directory, 'default')

    ]

    # Output directory for JSON and CSV files

    output\_directory = current\_directory

    # Parse the hives to JSON and CSV

    parse\_hives\_to\_json\_and\_csv(hive\_paths, output\_directory)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

# References

**There are no sources in the current document.**