WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

TraceQuest: A System Information and Artifact Collection Tool

Kaushal Kumar Singh 1st.

DY Patil International University, Akurdi, Pune 411044

A B S T R A C T

TraceQuest is a cutting-edge digital forensic tool specifically engineered for live system analysis and incident responses on Windows operating systems. By automating the acquisition of critical system artifacts, TraceQuest enables the rapid assessment of system compromise and user activity. The tool performs an in-depth collection of installed application metadata, including application names, install dates, publisher details, version numbers, and uninstall strings, as well as support and update URLS. It also conducts advanced USB device forensics by leveraging registry analysis to enumerate the connected devices and resolve localised strings for precise device identification. Furthermore, TraceQuest features robust Google Chrome history extraction, ensuring data integrity through temporary database copies to prevent data corruption during live acquisitions. A key feature is the capability to operate in a live forensic environment, thereby minimising the impact on the system while maximising data integrity. TraceQuest provides a user-friendly interface for efficient data retrieval, complemented by comprehensive error handling and detailed logging, making it a valuable asset for digital forensic investigators, incident responders, and system administrators who need to assess and respond to security incidents. The extracted data can be exported to PDF reports, streamlined investigations, and reported workflows.

Keywords: Artifacts, Evidence, Event Logs, Windows Functions, Windows Registry, Forensics.

1. INTRODUCTION

Digital forensics is a discipline that is becoming increasingly vital, driven by the continuous rise in the use of digital devices and the resulting surge in cybercrime. The capability to quickly and accurately obtain system data and artifacts is essential for effective incident response, investigation, and threat analysis. Old-fashioned forensic techniques are usually based on the offline examination of disk images, which can take a long time and may be impractical in situations where prompt evaluation of an active system is needed. Live forensics, however, allows investigators to analyse an active system to look for malicious processes, network activity, and the temporary information that could be deleted during a shutdown or imaging procedure.

In this research, I present TraceQuest, a new digital forensic tool that aims to simplify the acquisition of critical system artifacts from live Windows systems. TraceQuest is centred on automating the acquisition of installed application metadata, comprehensive USB device information, and Google Chrome browsing history while ensuring data integrity and reducing the target system's impact.

This utility uses direct registry analysis and temporary database copies to guarantee forensic accuracy and soundness. The gap between extensive forensic analysis and the need for quick incident evaluation gives system administrators and investigators a simple and effective way of comprehending system activity and user behaviour. This article outlines the design, implementation, and functionality of TraceQuest, emphasising its potential to improve live forensic investigations and incident response. This utility uses direct registry analysis and temporary database copies to guarantee forensic accuracy and soundness. The gap between extensive forensic analysis and the need for quick incident evaluation gives system administrators and investigators a simple and effective way of comprehending system activity and user behaviour. This article outlines the design, implementation, and functionality of TraceQuest, emphasising its potential to improve live forensic investigations and incident response processes

* *Kaushl Kumar Singh.* Tel.: 7030304288
E-mail address: 20230804030@dypiu.ac.in

2. BASICS OF WINDOWS FORENSICS

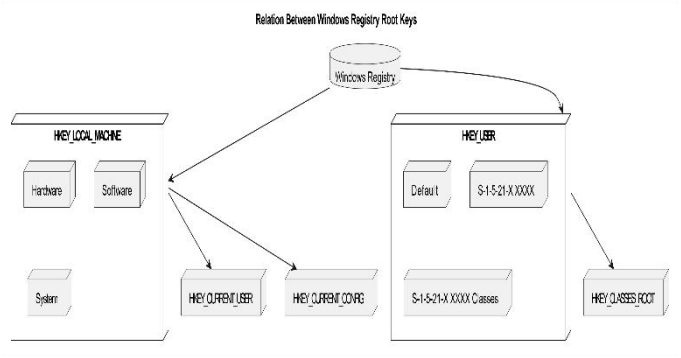
Windows forensic analysis focuses on building in-depth digital forensics knowledge of Microsoft Windows operating systems. You can't protect what you don't know about, and understanding forensics capabilities and artifacts is a core component of information security. To recover, analyze and authenticate forensics data on Windows systems, track particular activity on your network and organize findings for use in incident response or internal investigations. Windows is silently recording an unbelievable amount of data about users. It also stores each movement in the system.

A. Windows Event Logs

Logs are records of events that happen in your computer, either by a person or by running processes. They help you to track what happened and troubleshoot the problems. In any event of Windows investigation, Windows event logs serve as the primary source of evidence as the operating system logs every system activity. Windows event logs can help an investigator draw timeline-based logging information and discover artifacts. An event log contains an abundance of data, which enables for investigation of any abnormal cases or unauthorized access of the system. The investigator can acquire that file in text, and then it will be analyzed using third-party tools like QRadar.

B. Windows Registry

For a forensic analyst, the registry is a treasure box of information. It is the database that contains the default settings, user-defined and system-defined settings in a Windows computer. The registry serves as the repository, monitoring, observing and recording the activities performed by the user in the computer. The data is stored in the main folders in a tree-like structure, which is called Hive, and its sub-folders are called Keys and Sub-keys, where each component's configuration is stored and denoted as Values.



3. RELATED WORK

2. A. Current tools used to investigate entities, analyze logs and parse registry data

1)Manage Engine Event Log Analyzer, NXLog Agent

2) Windows Live Response

B. Limitations of the Current tools used while parsing the registry, investigating entities and analysing logs

1) Some tools can only pool standard events like system security applications and setup logs from Windows machines, and don't have the capabilities to pool application and service logs like systems.
2) Manage Engine and Event log analysis can collect log-based Windows event channels and can only collect the latest logs generated on the system post-installation.
3) Manage Engine, Registry Recon and NXLog Enterprise Edition are paid software for use.

C. The author's initial Windows Forensic tools play a crucial role in collecting and preserving evidence during digital investigations. One notable contribution to this field is the Python-based tool described in the paper, **"Forensic Evidence Collection From Windows Host Using Python Based Tool"**, authored by **Mr. Amit Hariyani, Dr. Jaimin Undavia, Dr. Nilay Vaidya** and **Dr. Atul Patel.** This tool introduces an efficient and automated way to collect critical forensic artifacts from Windows systems. While it addresses key challenges in artifact collection, it leaves room for improvement in terms of scope, usability, and modularity.

Feature	Python-Based Tool (Referenced Paper)	TraceQuest
Artifact Coverage	Focused primarily on PowerShell events, Sysmon logs, USB history, and Chrome history.	Expands artifact collection to include network configurations, startup programs, installed applications, and detailed logon sessions.
System Information Collection	Limited system metadata extraction (e.g., OS, date/time).	Comprehensive system-level data, including motherboard, BIOS, CPU, physical memory, and logical disk information.
Network Data Analysis	Not supported.	Includes active network connections, DNS configuration, network adapter details, and routing tables.
Process and Software Insights	Limited.	Provides detailed insights into running processes, task lists, and startup programs.
Interface	Command-line based, requiring higher technical expertise.	User-friendly GUI with categorized buttons and real-time output visualization using Tkinter.
Reporting	Generates text-based outputs.	Automates PDF report generation with customizable examiner details and timestamps.

4. PROBLEM SUMMARY

Most of the time, a forensics/incident investigator faces below problems/challenges.
1) The required artifacts are not available on the system
2) If artifacts are available, the investigator doesn't know where to check and how to extract specific artifacts, which can then be investigated in a lab environment
3) Some tools don't have the capabilities to extract specific artifacts, which leads to collecting all the data, which is time-consuming as well as requires more storage for collection and time for investigation
4)Forensics involves gathering data from various sources (system, network, registry, etc.). TraceQuest solves the problem of manually collecting this data by automating the process through WMI queries, registry analysis functions, and file system interactions.

5. METHODOLOGY

5.1 System Architecture

The tool is implemented in Python, leveraging libraries such as WMI (Windows Management Instrumentation), winreg (Windows Registry API), sqlite3 (SQLite database access), and tkinter (GUI framework). This architecture allows for direct system access while maintaining a low resource footprint on the target system.
The tool's components are organised as follows:

- GUI Interface: Provides user interaction and result display
- Collection Modules: Individual functions for specific artifact collection
- Data Processing: Parsing and formatting collected data
- Reporting: Generating structured reports of findings

5.2 USB Device History Collection

USB device collection methodology extracts information from the Windows registry, specifically targeting the SYSTEM\CurrentControlSet\Enum\USB registry path. This location contains detailed information about USB devices that have been connected to the system, including:

- Device identifiers
- Connection histories
- Hardware details

The collection process accesses the registry using Python's winreg module, traversing the relevant keys to extract device information while handling potential permission issues and access errors.

5.3 Web Browser History Collection

For browser history collection, our tool focuses on Google Chrome, one of the most widely used browsers. The collection process involves:

1. Locating the Chrome history database in the user profile
2. Creating a temporary copy to avoid database locks
3. Querying the SQLite database for URL, title, and timestamp information
4. Formatting the results for analysis

The tool handles potential issues such as corrupt databases, access permissions and ensures proper cleanup of temporary files after extraction.

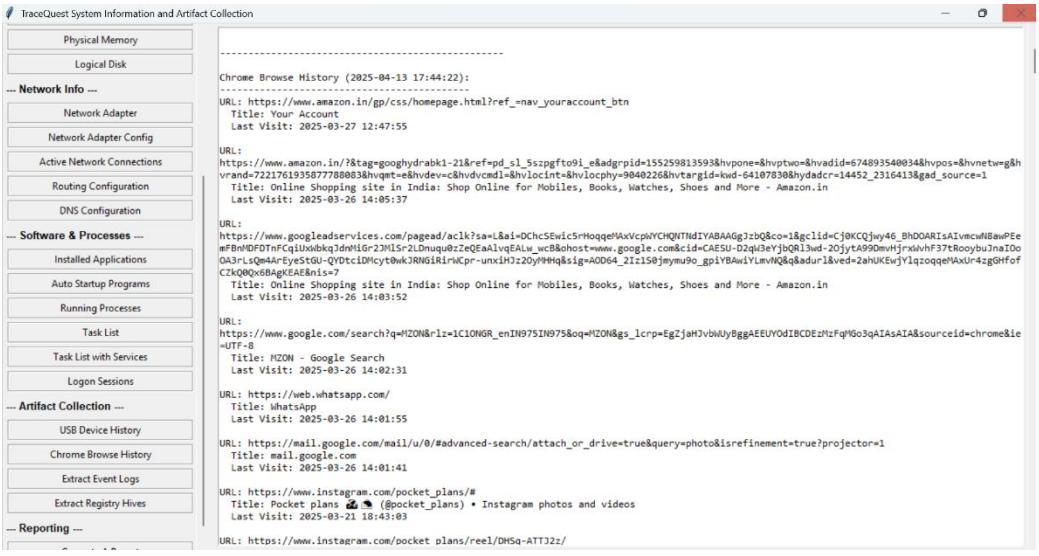


Fig 5.3.1 Chrome Browser's History

5.4 System Information Gathering

Beyond USB and browser artifacts, the tool collects comprehensive system information using WMI queries and command-line utilities:

- Hardware information (motherboard, BIOS, processor)
- Operating system details
- Network configuration
- Running processes and services
- Installed applications
- User account information

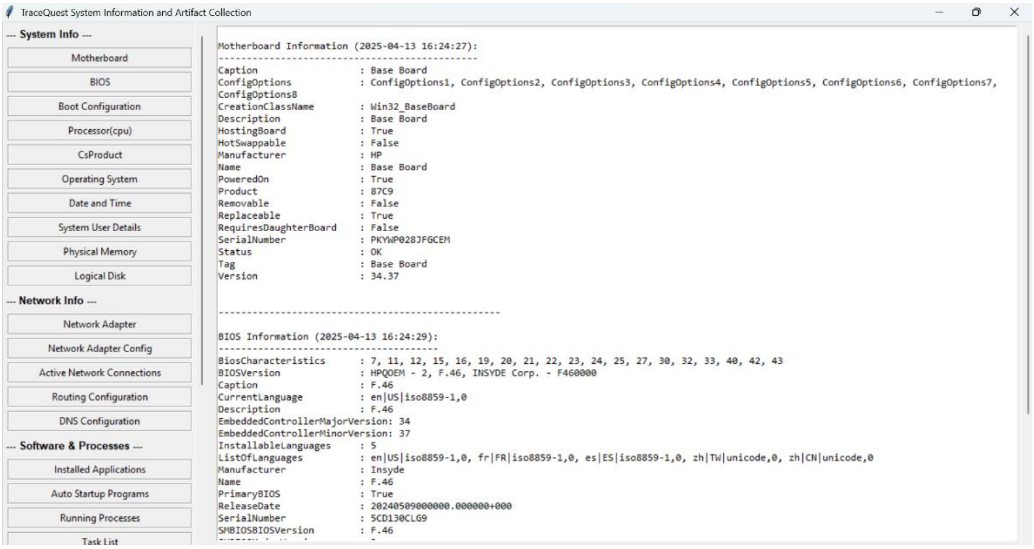


Fig 5.4.1 System Information

5.5 Registry and Event Log Export

To facilitate deeper forensic analysis, the tool includes functionality to export Windows registry hives and event logs:

- Registry hives (SYSTEM, SOFTWARE, SAM, SECURITY)
- Windows event logs (System, Application, Security)

These exports are saved with timestamps and proper documentation to maintain the chain of custody.

Fig 5.4.2 Log File Extraction

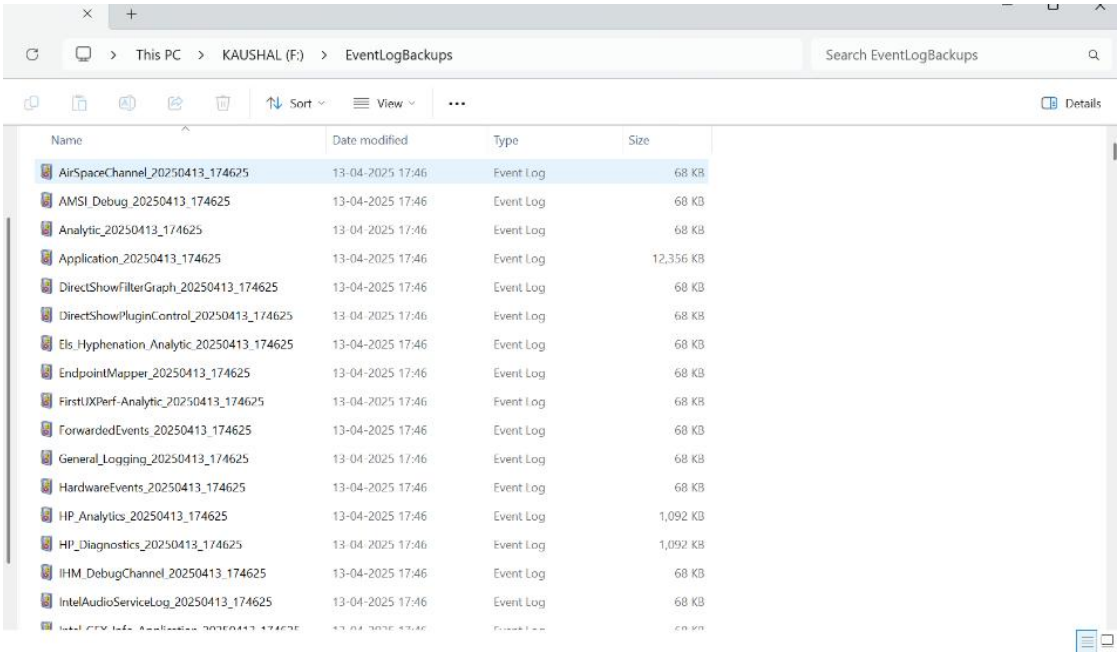


Fig 5.4.2 Extracted Log Files

6. IMPLEMENTATION

6.1 User Interface

The tool employs a two-panel interface:

- Left panel: Categorized buttons for different collection functions
- Right panel: Display area for results and collection status

The interface includes scrollable sections to accommodate various screen sizes and large amounts of collected data. Error handling is integrated to provide feedback on collection status and any issues encountered.

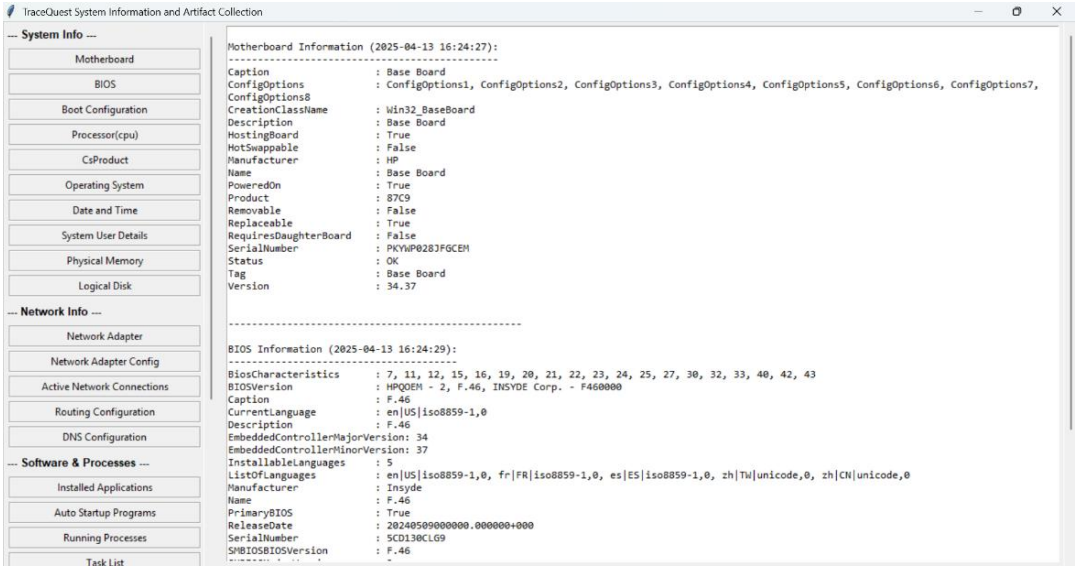


Fig 6.1.1 User Interface

6.2 Key Implementation Features

The core system handles basic operations and provides foundational services to the specialized modules. Key components include:

```
import os
import winreg
import sqlite3
import shutil
import tempfile
from datetime import datetime
import tkinter as tk
from tkinter import ttk
from tkinter import simpledialog, messagebox, filedialog
import wmi
import re
import subprocess
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from reportlab.platypus import Paragraph, Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.enums import TA_LEFT, TA_CENTER
from reportlab.lib import colors
```

The implementation leverages several key Python libraries:

winreg **for** Registry access
wmi **for** Windows Management Instrumentation queries
sqlite3 **for** database interactions
subprocess **for** executing system commands
Reportlab **for** PDF report generation

The USB device information collection function demonstrates the tool's approach to registry access and error handling. Similar methodologies are applied to other collection functions.

```
def gather_usb_device_info():
    """ Gathers USB device information from the registry and returns it."""
    usb_info = []
    try:
        registry_path = r"SYSTEM\CurrentControlSet\Enum\USB"
        registry = winreg.OpenKey(winreg.HKEY_LOCAL_MACHINE, registry_path)
        for i in range(winreg.QueryInfoKey(registry)[0]):
            try:
                sub_key_name = winreg.EnumKey(registry, i)
                usb_info.append(f"Device Key Root: {sub_key_name}")
            except OSError:
                continue  # Handle potential permission issues with subkeys
        winreg.CloseKey(registry)
        return usb_info if usb_info else ["No USB device information found in registry path."]
    except Exception as e:
        return [f"[!] Error gathering USB Device Information: {e}"]
```

For Chrome history collection:

```
def collect_chrome_history():
    """Copies Chrome history DB, collects history, and returns it."""
    history_data = []
    try:
```

```
        home_dir = os.path.expanduser("~")
        db_path_original = os.path.join(
            home_dir,           r"AppData\Local\Google\Chrome\User Data\Default\History"
        )
    if not os.path.exists(db_path_original):
            return "[!] Chrome history database not found. Is Chrome installed?"
    # Create a temporary copy to avoid database lock issues
        temp_dir = tempfile.gettempdir()
        temp_db_path = os.path.join(temp_dir, "chrome_history_temp.db")
        shutil.copy2(db_path_original, temp_db_path)
        conn = sqlite3.connect(temp_db_path)
        cursor = conn.cursor()
        cursor.execute(
            "SELECT url, title, datetime(last_visit_time/1000000-11644473600, 'unixepoch', 'localtime') as last_visit FROM urls ORDER BY last_visit_time
DESC"
        )
        rows = cursor.fetchall()
        conn.close()
        for row in rows:
            history_data.append({
                "url": row[0],
                "title": row[1] if row[1] else "No Title",
                "last_visit": row[2]
            })
        # Clean up temporary file
        if os.path.exists(temp_db_path):
os.remove(temp_db_path)
        return history_data
    except Exception as e:
        return f"[!] Error in collecting Chrome History: {e}"
```

This implementation demonstrates the handling of database connections, SQL queries for artifact extraction, and proper resource cleanup.

6.3 Report Generation

The tool generates structured PDF reports using ReportLab, preserving the formatting and organization of collected data:

1. Examiner identification for chain of custody
2. Timestamps for each collection activity
3. Categorized presentation of findings
4. Proper pagination and formatting.

7. EVALUATION

7.1 Performance Analysis

The tool was evaluated on multiple Windows systems (Windows 10 and Windows 11) with varying hardware configurations. Key performance metrics included:

- Collection time for each artifact type
- Resource utilization during collection
- Accuracy of collected data compared to manual extraction

Testing revealed efficient performance, with most collection operations completing within seconds to minutes, depending on system size and artifact volume.

7.2 Data Accuracy

Validation testing compared the tool's output with manually extracted artifacts using native Windows tools and commercial forensic solutions. The comparison confirmed the accuracy of:

- USB device identifiers and connection history
- Chrome browsing history entries and timestamps
- System configuration details
- Registry and event log exports

7.3 Use Case Scenarios

The tool was applied to several forensic scenarios:

1. **Data Exfiltration Investigation**: Identifying USB devices connected during suspicious periods
2. **User Activity Analysis**: Reconstructing web browsing activity in insider threat cases
3. **System Compromise Assessment**: Gathering system configuration and running processes for malware analysis

In each scenario, the tool successfully collected relevant artifacts for further analysis.

8. Limitations and Future Work

While the tool provides valuable forensic capabilities, several limitations exist:

1. Limited to locally accessible systems (no remote collection)
2. Currently only supports Chrome browser history (not Firefox, Edge, etc.)
3. USB device history collection could be expanded to include more detailed connection timestamps

Future work will address these limitations by:

1. Adding support for additional browsers
2. Enhancing USB artifact collection with connection timestamps from event logs
3. Implementing basic analysis functionality for common investigation types
4. Adding support for remote system collection

9. CONCLUSION

The enhanced forensic tool presented in this paper provides digital investigators with an effective means of collecting critical artifacts from Windows systems. By focusing on USB device history and web browsing activities while maintaining comprehensive system information collection, the tool addresses common investigative needs.

The open-source nature of the tool allows for customization and validation, important factors in forensic tool acceptance. The structured reporting capabilities ensure proper documentation for evidentiary purposes.

This work contributes to the digital forensics field by demonstrating practical implementations of forensic collection methodologies in a unified, accessible tool that can be deployed in real-world investigations.

10. ACKNOWLEDGEMENT

This Python-based tool is a part of **Forensic Artifacts Collection from Windows Host.** The objective of the consecutive article is to examine Windows file systems, registry, event logs, system monitoring, web browser history, and active network connections for analysis to help the investigator. Windows artifacts are very crucial evidence that is useful for the investigation of USB registries, registry runs, etc. Though this tool is currently built to collect windows registry, event logs, browser history, and network connections but additional capabilities can be added to this by leveraging python modules such as gathering additional details from windows host like recycle bin, running processes, and other event logs; forwarding data to centralized servers and streaming real-time events to SIEM platform. Also collecting data from Linux/Unix and Mac OS-based systems.

REFERENCES

[1] Hariyani, A., & Undaiva, J. (2022). Forensic Evidence Collection from Windows Host Using Python-Based Tool. In 2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA). DOI: 10.1109/ICCCMLA56841.2022.9989295.
[2] B. Carrier, "File System Forensic Analysis," Addison-Wesley Professional, 2005.
[3] H. Carvey, "Windows Forensic Analysis Toolkit: Advanced Analysis Techniques for Windows 10," Syngress, 2016.
[4] E. Casey, "Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet," Academic Press, 2011.
[5] OpenText, "EnCase Forensic," [Online]. Available: https://www.opentext.com/products/encase-forensic
[6] AccessData, "Forensic Toolkit (FTK)," [Online]. Available: https://www.exterro.com/forensic-toolkit
[7] N. Agrawal and N. Giramkar, "Forensic Evidence Collection From Windows Host Using Python Based Tool," In 2022 Fourth International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 1208-1213, IEEE, 2022.
[8] J. Collie, "The Windows registry as a forensic resource," Digital Investigation, vol. 10, no. 2, pp. 157-165, 2013.
[9] S. Lim, B. Ahn, and K. Lee, "Windows Registry Analysis for MAC Times," International Conference on Information Security and Assurance, pp. 574-578, 2008.
[10] S. Garfinkel and D. Cox, "Finding and Archiving the Internet Footprint," Digital Lives Research Conference, 2009.