A Practical Study on Automating Windows Digital Forensics for Faster Incident Response

Shahid Shaikh
Security Researcher
Guru Nanak Khalsa College, Mumbai MH - 19

Abstract : Windows digital forensics is essential for finding out what really happened during a security incident, but in practice the work is often slow and fragmented across many tools and manual steps. This paper reports on a practical study of an automated framework that focuses on speeding up the early stages of Windows incident response. The framework automatically collects important artifacts, including event logs, running processes, network connections, browser history, USB device activity and disk encryption status, and then arranges them into a single, time-ordered view of system and user actions. The study compares a typical semi-manual workflow with the automated approach on several Windows machines, looking at the time needed to build an initial picture of the incident and the completeness of the evidence collected. The observations indicate that automation can cut the time to build a usable timeline while still keeping enough detail for deeper manual investigation where required. Overall, the work shows how a carefully designed automated workflow can help responders answer key questions such as what happened, when it happened and which artifacts support those conclusions, leading to more efficient Windows incident response in real environments.

Keywords: Windows digital forensics, automation, incident response, forensic artifacts, timeline reconstruction, evidence collection

1. INTRODUCTION

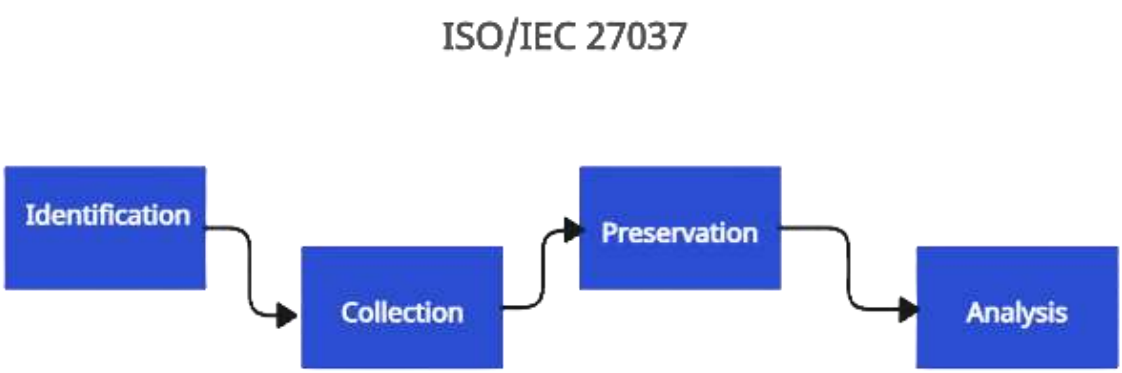
Windows systems generate a large amount of digital evidence during normal use, including logs, process data, browser history, USB activity and network traces. During a security incident, this information is vital for understanding what happened on a machine, but it is usually spread across many different locations and tools. Investigators often need to run several utilities, export files in different formats and manually combine results before they can even start real analysis.

This manual and fragmented workflow slows down incident response and increases the chance of overlooking important artifacts, especially in the early stages of an investigation. Repeated tasks such as collecting standard logs, parsing browser history or listing installed software are time-consuming but predictable, and therefore good candidates for automation. A practical automated system that can gather common Windows artifacts and arrange them into a clear, time-based view has the potential to reduce effort, shorten response times and make investigations more consistent.

The work presented in this paper explores such an approach. It focuses on automating the collection of key Windows forensic artifacts and rebuilding a readable activity timeline to support faster and more reliable incident response, while still allowing investigators to review the underlying evidence when needed.

Digital forensics, in general, is concerned with identifying, collecting, preserving and analysing electronic evidence in a way that is technically sound and legally defensible. Windows forensics is a sub-area that concentrates on artifacts created by the Windows operating system and applications, such as event logs, registry keys, user profiles, browser data and file system metadata. Standards like ISO/IEC 27037 give high-level guidance for handling digital evidence and describe the forensic process as a sequence of activities: Identification, Collection, Preservation and Analysis. In the context of Windows systems, this means first recognising which artifacts are relevant to a case, then acquiring them in a controlled manner, protecting their integrity and finally examining them to reconstruct user and system activity.

In this paper, the proposed automated framework is aligned with these stages. It helps with Identification by pointing to common Windows evidence sources, automates the Collection of those artifacts, applies consistent procedures to support Preservation and presents the results in a form that supports structured Analysis, such as timeline views used during incident response.



2. LITERATURE REVIEW.

Automation in digital forensics has developed over time from simple scripts for collecting logs to more advanced frameworks that support acquisition, parsing and visualisation of evidence. For Windows systems, investigators have long relied on a mixture of commercial tools, open-source frameworks and custom utilities to examine disks, memory and live system artifacts. While these tools provide strong technical capabilities, many studies highlight that the overall workflow for incident response is still fragmented and labour-intensive, especially when analysts need to combine outputs from several sources and produce a clear activity timeline.

2.1 Existing Digital Forensic Tools

Commercial forensic suites such as EnCase and FTK offer full workflows for imaging, analysis and reporting on Windows machines, but they are costly and often require dedicated training to use effectively. open-source tools like Autopsy and The Sleuth Kit are widely used in academia and smaller organisations for disk and file-system analysis. Memory forensics frameworks including Volatility and Rekall allow detailed inspection of running processes, injected code and network connections captured from RAM on Windows hosts. specialised utilities exist for individual artifact types, for example registry viewers, browser history parsers and Windows event log analysis tools, each with its own interface and export format.

2.2 Automation and Triage Approaches

Research on forensic triage proposes scripted collectors that gather common Windows artifacts such as logs, processes, network information and browser data into a single package to support rapid first-look investigations. some frameworks automatically parse collected artifacts into structured databases, enabling faster search, filtering and basic reporting without manual conversion. timeline-based tools focus on ordering artifacts chronologically to help investigators reconstruct user activity around an incident, though many are limited to specific sources like file system timestamps or a subset of logs.

2.3 Gaps Identified in Prior Work

Investigations often require analysts to move between multiple tools and formats, which increases effort and makes it harder to maintain a clear picture of events across a Windows system.

Many powerful tools have a steep learning curve, with complex interfaces or command-line usage that can slow down less experienced responders and small teams. existing automation efforts are frequently tied to particular environments or focus on narrow artifact sets, and there are relatively few lightweight frameworks that automatically collect a broad range of Windows artifacts and present them through an integrated, incident-oriented timeline view.

The present study builds on this body of work by examining a practical framework that combines multi-source Windows artifact collection with unified timeline reconstruction aimed at faster and more accessible incident response.

3. PROBLEM STATEMENT AND OBJECTIVES.

3.1 Problem Statement

Windows incident investigations usually begin with basic but time-consuming tasks: collecting event logs, listing running processes, checking network connections, reviewing browser history, identifying connected USB devices and confirming disk encryption status. Today, many teams still perform these activities using separate tools and manual steps, exporting data in different formats and then trying to combine the results into a clear picture. This fragmented workflow slows down incident response, increases the chance of missing important artifacts and makes it harder for less experienced analysts to follow good forensic practice. There is a practical need for a simple, repeatable way to automate these common tasks on Windows systems and to present the results in a form that directly supports understanding what happened during an incident.

3.2 Objectives of the Study

The main objectives of this study are:

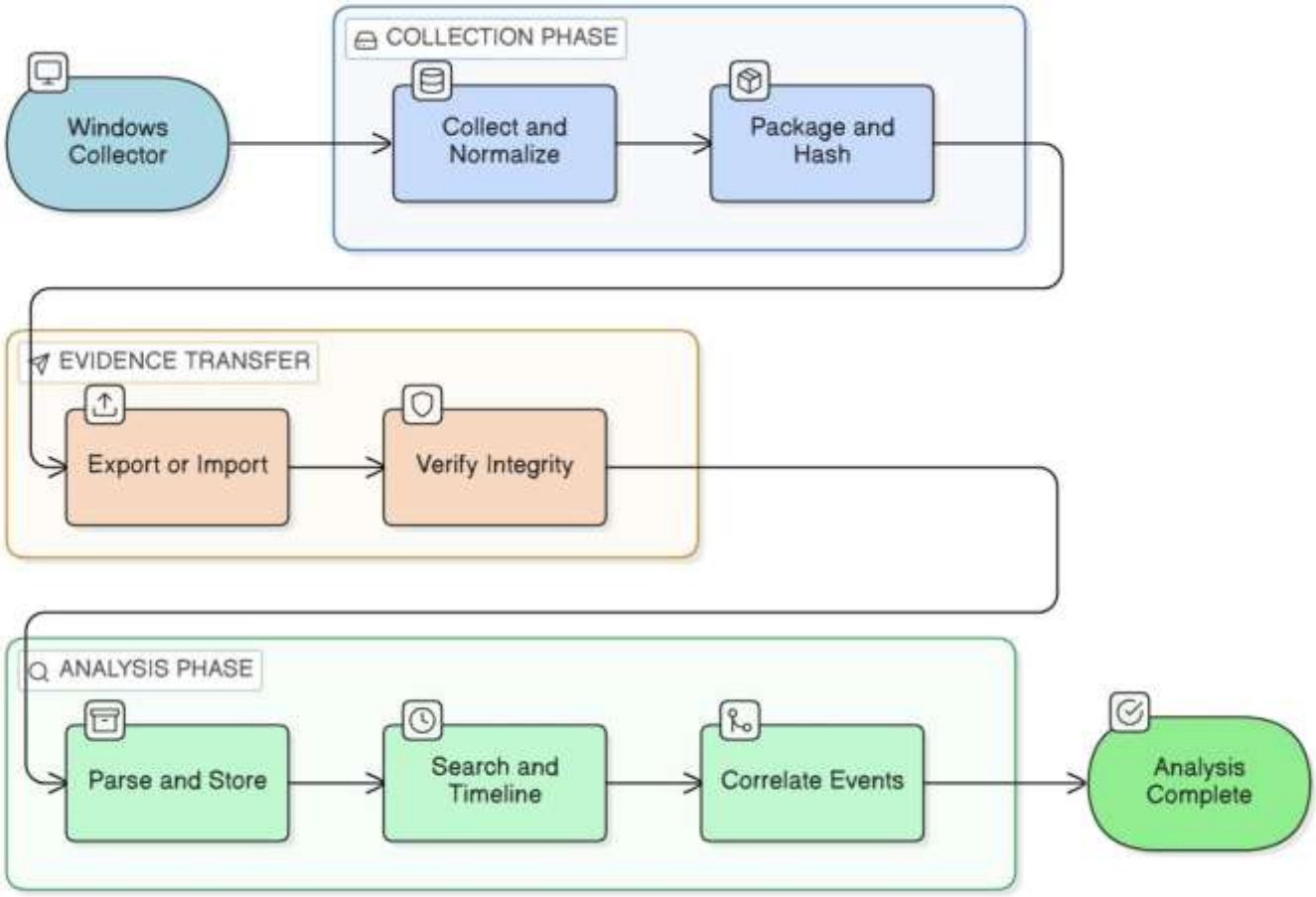
- To design and describe an automated framework that collects key Windows forensic artifacts in a consistent and repeatable way.
- To align the collection and handling of these artifacts with standard forensic stages such as identification, collection, preservation and analysis.
- To generate a unified, time-ordered view of user and system activity by correlating artifacts from multiple sources into a single timeline.
- To compare the automated approach with a typical semi-manual workflow in terms of effort, time to obtain first investigative findings and completeness of collected evidence.
- To evaluate how such automation can support faster and more reliable Windows incident response while still allowing detailed manual analysis when required.

4. METHODOLOGY

4.1 Overall Architecture

The proposed system follows a simple two-part architecture: a Windows "collector" component and a separate "analyzer" component. The collector runs on or against the target Windows machine and is responsible for identifying and extracting the main forensic artifacts in a repeatable way. It does not try to perform heavy analysis on the live system. Instead, it gathers data, normalises it where possible and packages it for later use. The analyzer is a separate application that imports this package, stores the artifacts in a structured form and provides views such as search, filtering and timelines to support incident response.

The design goal is to keep the collector as lightweight and predictable as possible so that it can be executed quickly during an investigation with minimal interaction. The analyzer can then be more feature-rich and can run on a workstation or server where investigators have time and resources to explore the evidence. This separation also supports the forensic principle of preserving the original system as much as possible while still obtaining the information needed for analysis.

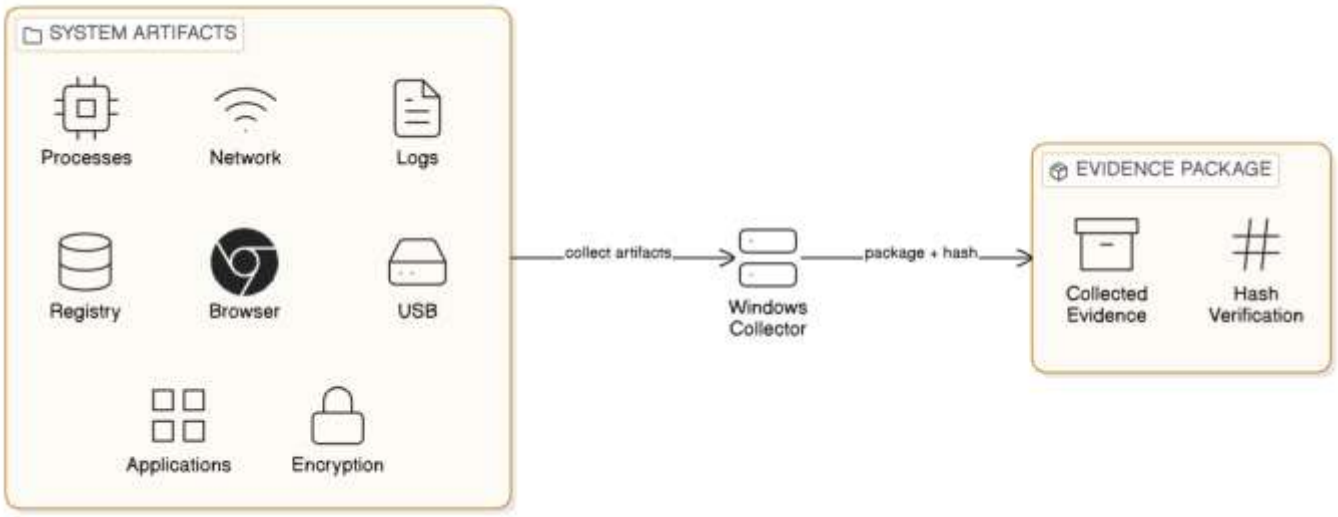


4.2 Windows Collector Component

The collector component is designed to run as a single executable or script on a Windows system with sufficient privileges. Its main task is to gather a defined set of artifacts that are commonly used in incident response. In the current design, the collector focuses on the following categories:

- **System and process information:** snapshots of running processes, services and scheduled tasks, including process names, paths, start times and parent-child relationships.
- **Network artifacts:** active connections, listening ports, basic firewall configuration and information about Wi-Fi profiles or other network interfaces.
- **Event logs:** selected Windows event log channels related to security, system events, logon activity and PowerShell usage.
- **Registry keys:** key areas of the registry that hold information about persistence mechanisms, recently used files, installed software and USB devices.
- **Browser data:** history, downloads and other relevant records from major browsers installed on the system, focusing on URLs, visit times and associated profiles.
- **USB and device history:** information about removable storage devices and other hardware that has been connected to the system over time.
- **Application and executable details:** lists of installed applications and the results of file system scans for executables in common locations, including basic metadata and hash values.
- **Disk encryption and security status:** details on BitLocker or other encryption technologies, as well as summary information about antivirus and firewall status.

For each category, the collector uses standard Windows interfaces where possible, such as built-in command-line tools, registry queries and log APIs, to reduce the need for additional dependencies. The collected data is stored in structured formats such as JSON or CSV and then combined into a single evidence package with integrity checksums so that later analysis can verify that the data has not been altered.



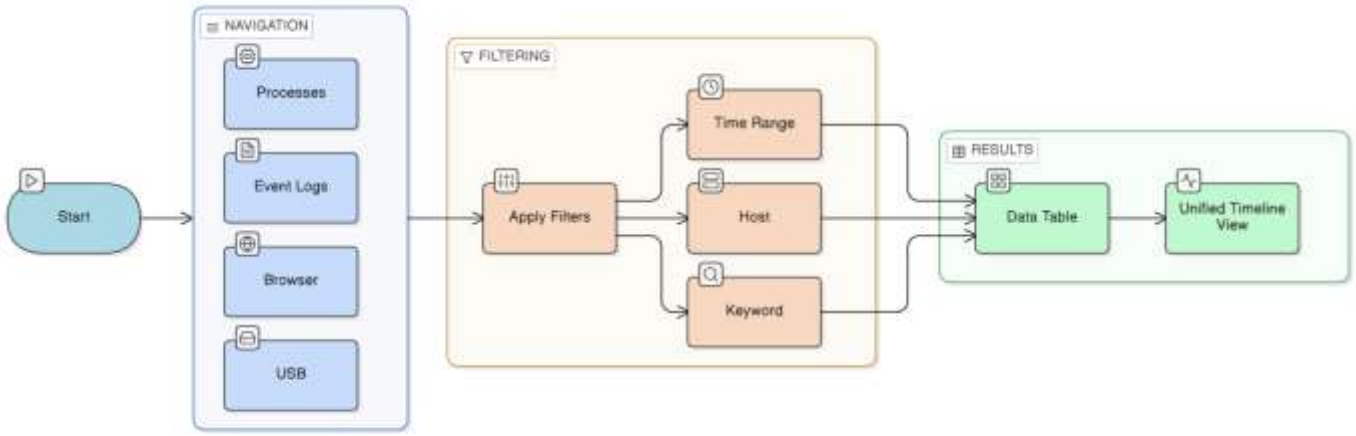
4.3 Analyzer Component

The analyzer component runs on an analysis system and is responsible for importing the evidence package, parsing the contained files and storing them in a local database. Each artifact type is mapped to an internal model so that investigators can query and filter data across multiple sources. For example, process entries, event log records and browser history items can all be stored with their timestamps, user context and host information.

The analyzer provides basic views such as:

- tables for each artifact type, allowing filtering by time range, host, username or keyword;
- quick search across selected fields to find events related to a particular process name, URL or device;
- summary panels that highlight counts of artifacts or potential indicators of suspicious activity.

These views are not intended to replace full forensic suites, but to make it easier to perform early triage and to identify which artifacts deserve deeper manual examination.



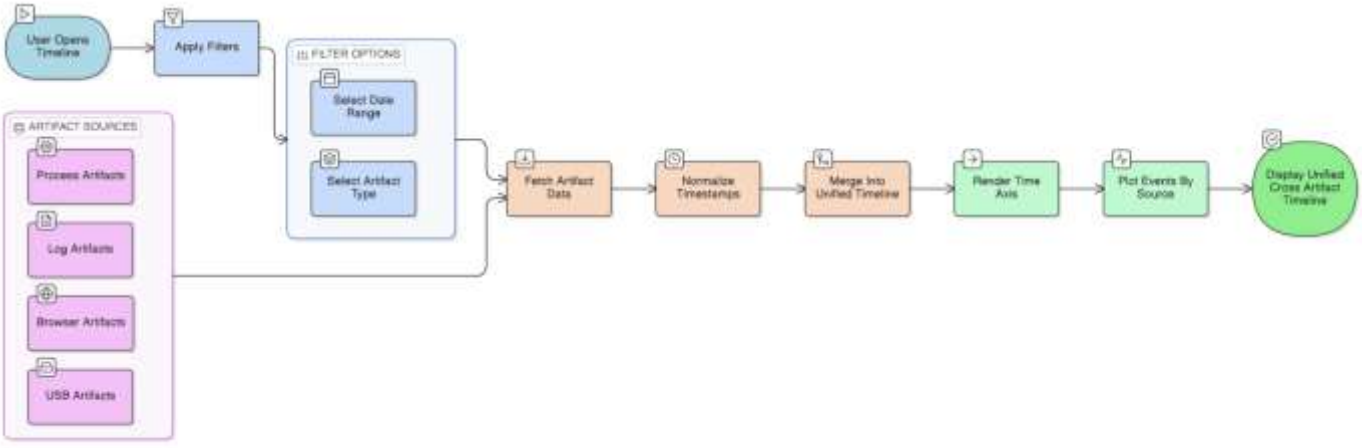
4.4 Timeline Generation and Correlation

A key feature of the proposed system is the ability to reconstruct a unified timeline of activity across different artifact types. Once artifacts are imported into the analyzer, each record that contains a timestamp is normalised to a common time format, taking into account time zones and any known system offsets. The system then builds a merged list of events ordered by time, with each entry recording at least the following fields:

- timestamp (normalised),
- source type (for example, event log, process, browser, USB),
- short description,
- host and, where possible, user account.

From this merged list, the analyzer can group events by day, hour or incident window. Investigators can scroll through the timeline to see, for example, when a new executable appeared on disk, which user was logged in at that time, what network connections were active and which websites were visited before and after the event. Simple filters allow the timeline to be focused on a particular process name, IP address, URL or device identifier.

To support correlation, the system uses shared fields such as process identifiers, filenames, IP addresses and user names to highlight potential links between events. For instance, if an executable with a particular hash is seen in the file system scan, and the same path appears in an event log entry or network connection record, the analyzer can group or mark these entries in the timeline. This is not intended to replace full correlation engines, but it helps the investigator notice relationships that might otherwise be missed in a manual review.



4.5 Alignment with Forensic Process

The overall methodology is designed to stay consistent with the stages suggested in ISO/IEC 27037: Identification, Collection, Preservation and Analysis. During Identification, the system defines and documents which Windows artifacts are in scope. The collector performs the Collection step by automatically acquiring these artifacts using controlled procedures. Preservation is supported through the use of read-only or minimally intrusive collection methods and by packaging the data with hash values to detect any changes. Analysis takes place mainly in the analyzer, where investigators can review the artifacts, explore the unified timeline and generate observations that can be used in incident reports.

By following this structure, the proposed framework aims to provide not only technical efficiency but also a clear and defensible workflow that can be explained to management, auditors or legal stakeholders when necessary.

5. EXPERIMENTAL SETUP AND RESULTS

5.1 Test Environment

To evaluate the proposed framework in a realistic but controlled way, a small lab environment was created using common Windows configurations. Three types of machines were prepared:

- **Machine A** – Office laptop: Windows 10 Pro, used for web browsing, document work and email.
- **Machine B** – Developer workstation: Windows 11 Pro, with development tools, local web servers and frequent PowerShell usage.
- **Machine C** – Test VM: Fresh Windows 10 installation used to replay specific incident scenarios such as malware execution and suspicious USB usage.

Each machine was connected to a local network with basic internet access and standard endpoint protection enabled. On every system, normal user activity was simulated for several days: browsing websites, downloading files, installing and uninstalling applications, copying data to and from USB drives and running simple scripts. On Machine C, additional actions were performed to imitate incident conditions, such as running an unknown executable from a download folder, disabling antivirus temporarily and connecting an unfamiliar USB device.

5.2 Evaluation Procedure

The evaluation compared two workflows for performing an initial forensic review after an incident alert:

1. Semi-manual workflow:

- Investigator runs a set of standard tools and commands on the Windows host, for example exporting event logs, listing processes, capturing network connections, dumping registry keys and manually extracting browser history.
- Collected files are copied to an analysis machine, where separate viewers are used to open logs, registry exports and browser databases.
- The investigator builds a basic incident timeline by reading timestamps and taking notes.

2. Automated workflow (proposed system):

- Investigator runs the Windows collector once with a predefined profile.
- The generated evidence package is imported into the analyzer.
- The investigator uses the integrated views and timeline to answer the same initial questions.

For each workflow and each machine, the following measures were recorded:

- time from “start of evidence collection” to “first usable timeline or summary view”;
- approximate number of separate tools or commands used;
- number of distinct artifact categories successfully collected (for example: processes, event logs, browser history, USB, applications, encryption, network);
- investigator’s subjective rating of effort on a simple scale (low, medium, high).

The same investigator performed both workflows to reduce variation due to skill differences, and a short written checklist was used to ensure that similar tasks were attempted in both cases.

5.3 Quantitative Results

Machine	Approach	Time to First Summary	Tools / Commands	Artifact Categories
Machine A (Office Laptop)	Semi-manual	~45 min	7–8	5–6 (logs, processes, network, browser, USB)
	Automated	**~15 min**	**2**	**7–8 (incl. applications, encryption)**
Machine B (Developer Workstation)	Semi-manual	~60 min	9–10	6–7
	Automated	**~20 min**	**2**	**8–9 (incl. PowerShell, network)**
Machine C (Test VM)	Semi-manual	~55 min	8–9	6–7
	Automated	**~18 min**	**2**	**8–9**

Across all machines, the automated workflow reduced the time to obtain a usable view of events by roughly one-half to two-thirds compared with the semi-manual approach. It also increased the number of artifact categories available in the same time window, because the collector could gather several sources in parallel without additional effort from the investigator.

5.4 Case Study Observations

On Machine C, a small case study was conducted around a simulated incident where a suspicious executable was downloaded and run from the user’s Downloads folder. In the semi-manual workflow, the investigator first examined the Downloads directory, then searched the event logs and process lists, and finally checked browser history and USB records. Building a coherent story required moving between multiple tools and cross-checking timestamps manually.

Using the automated workflow, the investigator started from the unified timeline view. Within a few minutes it was possible to see the browser download event, the creation of the executable on disk, the corresponding process start entry, related network connections and a later change in antivirus status on the same host. These events were grouped by time and host, making

it easier to describe the sequence without detailed correlation work. The investigator's effort rating for this case dropped from "high" in the semi-manual run to "low-medium" when using the automated framework.

5.5 Summary of Results

The experimental results suggest that the proposed automated framework can meaningfully reduce the time and effort required to reach first investigative findings in Windows incidents, while also improving the consistency of artifact coverage. The system does not replace deeper manual analysis with specialised tools, but it provides a faster and more structured starting point for incident response.

6. DISCUSSION

The experimental results show that automation can make a clear, practical difference in how quickly investigators can start understanding a Windows incident. In all three test machines, the time to reach a first usable timeline or summary view dropped from roughly an hour to around 15–20 minutes when the automated framework was used. In simple terms, this means that less time is spent on "plumbing" tasks like exporting logs and collecting lists, and more time can be used to think about what the evidence means. For teams working under time pressure, being able to cut the early collection and correlation phase by half or more is a meaningful improvement.

Another important observation is that the automated workflow increased the number of artifact categories that were available in the same time window. When working manually, investigators tend to focus on the sources they know best or have time for, such as event logs and browser history, and may skip others like application inventories or detailed USB history. The collector, in contrast, can gather a broader set of artifacts every time without extra effort, which makes the resulting timeline richer. This helps reduce blind spots and encourages a more consistent investigation process across different cases and analysts.

However, the experiments also highlight that automation does not remove the need for manual analysis. The framework provides a fast and structured starting point, but it still relies on the investigator to interpret what the events mean in context. For example, seeing that a process started or a USB device was connected does not automatically prove malicious intent. Analysts may still need to open specific logs in detail, run memory forensics with specialised tools or examine file contents with other utilities. In complex cases, deep manual work remains essential, and automation mainly serves to guide that work towards the most relevant areas.

There are also trade-offs in how much logic is built into the automated system. If the framework tries to label too many events as "suspicious" automatically, there is a risk of confusing users or creating false confidence. Keeping the design focused on reliable collection and clear presentation, while leaving higher-level judgement to the investigator, helps avoid this problem. At the same time, simple helpers—such as highlighting events related to a specific process, IP address or time window—can make the timeline more useful without turning it into a black box.

Finally, the study underscores the importance of usability. The reduction in the number of tools and commands used was almost as important as the time savings. Investigators only had to remember how to run the collector and how to navigate the analyzer, instead of juggling many different programs with different interfaces. For smaller teams or less experienced responders, this lowers the barrier to performing basic forensic checks and makes it more likely that incidents will be handled in a repeatable way that aligns with standard forensic stages.

7. CONCLUSION AND FUTURE SCOPE

This paper presented a practical study on automating key parts of Windows digital forensics to support faster incident response. The proposed framework combines a lightweight collector, which gathers common forensic artifacts from Windows systems, with an analyzer that imports these artifacts and reconstructs a unified, time-based view of activity. The experimental comparison with a typical semi-manual workflow on three different Windows machines showed that the automated approach can significantly reduce the time needed to reach first findings while at the same time increasing the range of artifacts available in that early phase. Instead of spending most of their effort on repetitive collection steps, investigators can move more quickly to interpreting what happened and planning next actions.

The results also highlight that automation is most effective when it is used to standardise collection and provide clear views, not to replace human judgement. The framework offers a richer and more consistent starting point, but detailed conclusions still depend on the investigator's experience and, in complex cases, on specialised tools for memory analysis, reverse engineering or deep log review. The work therefore supports, rather than replaces, traditional forensic practice.

In future, the framework can be extended in several directions. Additional artifact types could be integrated, such as logs from common business applications or popular communication tools, to give an even more complete picture of user activity. Basic scoring or tagging of events—based on rules, threat intelligence or simple anomaly detection—could help highlight parts of the timeline that deserve closer attention, while keeping the final interpretation in human hands. Support for other operating systems, such as Linux and macOS, would make the approach more widely applicable in mixed environments. Finally, improvements in reporting features, collaboration between analysts and secure remote collection would help move the system closer to real-world deployment in organisations that handle frequent incidents.

8. REFERENCES

[1] B. Carrier, File System Forensic Analysis. Addison-Wesley, 2005.
[2] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, 3rd ed. Academic Press, 2011.
[3] B. D. Carrier and E. H. Spafford, "Getting physical with the digital investigation process," International Journal of Digital Evidence, vol. 2, no. 2, 2003.
[4] A. Walters and N. Petroni, "Volatools: Integrating volatile memory forensics into the digital investigation process," IEEE Security & Privacy Workshops, 2007.
[5] S. Ligh, A. Case, J. Levy, and A. Walters, The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. Wiley, 2014.
[6] B. Schatz, "Bodyfarm: Building a corpus of digital forensics images for research and education," Digital Investigation, vol. 3, 2006.
[7] M. I. Husain and S. Kausar, "Automated digital forensic analysis of Windows artifacts," International Journal of Computer Applications, vol. 61, no. 20, 2013.
[8] M. Kovar, "Triage in digital forensics," in Proceedings of the 6th Annual Digital Forensics Research Workshop (DFRWS), 2006.
[9] ISO/IEC 27037:2012, Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence. International Organization for Standardization, 2012.
[10] B. Nelson, A. Phillips, and C. Steuart, Guide to Computer Forensics and Investigations, 6th ed. Cengage Learning, 2018.
[11] S. Garfinkel, "Digital forensics research: The next 10 years," Digital Investigation, vol. 7, pp. S64–S73, 2010.
[12] M. Quick and A. Choo, "Forensic collection and analysis of web browser artifacts," Journal of Network and Computer Applications, vol. 40, pp. 116–129, 2014.
[13] S. Chung and M. Kwan, "Investigating Windows event log for forensic analysis," International Journal of Digital Crime and Forensics, vol. 3, no. 2, pp. 52–67, 2011.
[14] R. L. Rivest, "The MD5 message-digest algorithm," RFC 1321, Internet Engineering Task Force, 1992.
[15] NIST, "Guide to Integrating Forensic Techniques into Incident Response," NIST Special Publication 800-86, U.S. National Institute of Standards and Technology, 2006.