DiagAnalyzer : User Behavior Analysis and Visualization Using Windows Diagnostics Logs

By:
Sungha Park and Sangjin Lee

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2022
Sept 28-30, 2022

Forensic Science International: Digital Investigation 43 (2022) 301450

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2022 APAC - Proceedings of the Second Annual DFRWS APAC

DiagAnalyzer: User behavior analysis and visualization using Windows Diagnostics logs

Sungha Park [a], Sangjin Lee [b, *]

[a] School of Law, Seoul National University, 1 Gwanak-ro, Gwanak-Gu, Seoul, South Korea
[b] School of Cybersecurity, Korea University, 145 Anam-ro, Seongbuk-Gu, Seoul, South Korea

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Windows Diagnostics
Eventtranscript.db
Windows forensics
Windows 10
Windows 11

ABSTRACT

Windows Diagnostics, which is used by default in Windows 10 and Windows 11, records basic device information as well as various detailed user activities of those who use Windows. Previously, there have been several preceding studies that attempted to apply diagnostics information to digital forensics analysis, but there have been no practical methods or publicly available tools to analyze data in relation to user behavior. Therefore, this paper analyzed how three representative activities (attaching and detaching USB storage devices, web browser activities, and wireless network activities) are recorded in Window Diagnostics. Furthermore, based on the analysis results, we developed DiagAnalyzer, which automatically analyzes the diagnostics event log and visualizes the user's behavior. Through the methodology and tool of this paper, the application of Windows Diagnostics deserves further attention as an important artifact in digital forensics investigation for Windows in the future.

1. Introduction

Microsoft records, collects, and manages Windows Diagnostics logs for fixing problems and improving product purposes of Windows 10 and Windows 11 (Microsofta). This information records major activities by the users, from basic device information to crash logs and device connection information, etc. Microsoft distributes Diagnostic Data Viewer (Microsoftb) to help users check and analyze the logs, but it is difficult to conduct an analysis beyond simply checking the event log as the tool only provides basic statistics and the users can check no more than the contents of a single event. If the user behavior is traced through additional analysis of the event logs recorded through Windows Diagnostics, it is highly likely to be applied for digital forensics investigations.

This paper analyzed three major behaviors of users (attaching and detaching USB storage devices, web browser activities, and wireless network activities) recorded in Windows Diagnostics. In addition, we developed *DiagAnalyzer*,[1] a tool that visualizes the analyzed behavior and outputs the analysis results as a report.

This paper will introduce the background information and preceding researches in Section 2. Section 3 will introduce detailed analysis results of three user behaviors. Section 4 will then introduce *DiagAnalyzer*, an automated Windows Diagnostics analysis and visualization tool, and the visualization results. Section 5 will discuss the contributions and limitations of this study, and directions for future research. Finally, section 6 is the conclusion, summarizing the main arguments of this paper.

2. Background information and related work

The Windows Diagnostics service is one of the default services used in Windows 7 and later Windows OS, and is run under the name of *DiagTrack*. The binary running the service is *diagtrack.dll* in the *C:\Windows\System32* directory. This binary is loaded by *utcsvc*, which is one of the core system services that is always served by the Windows NT family of operating systems. Windows 10 1709 and its later versions and Windows 11 store diagnostic logs in the *%ProgramData%\Microsoft\Diagnosis\EventTranscript* path as a SQLite3 type DB file, titled *EventTranscript.db*. Payload data of each event in the DB is stored in JSON format.

Diagnostics data is collected to keep Windows secure and up-to-date, troubleshoot problems, and improve products (Microsofta). Users can choose whether to collect only the required diagnostic data or the optional diagnostic data together. If the users decide to only collect the required diagnostic data, basic information such as device information, quality-related information and app

* Corresponding author.
E-mail addresses: ppeb15@snu.ac.kr (S. Park), sangjin@korea.ac.kr (S. Lee).
[1] https://github.com/L4wk3R/DiagAnalyzer.

https://doi.org/10.1016/j.fsidi.2022.301450

Table 1
The event_persisted table structure.

Column	Description
sid	The unique value of the device
timestamp	the timestamp the event was recorded
payload	Contains detailed data to be recorded in JSON format
full_event_name	The full name of the event
logging_binaryname	The name of the process that recorded the event

compatibility is the only information recorded. Conversely, if the users decide to collect the optional diagnostic data, specific information related to the use of Windows, such as connectivity, configuration data, service usage, and performance is collected together (Microsoftc).

Han et al. (2020) proposed a digital forensic analysis method using Windows Diagnostics and found the path and structure of the *.rbs file that record events in Windows Diagnostics. In addition, they revealed that OS, application, hardware, process execution, and partition information can be obtained through diagnostic logs, and suggested the possibility of incident response and utilization in IT audit. Currently, the storage path and format of the diagnostic log have changed as Windows has been updated, but the utilization method proposed by Han et al. (2020) is still valid today. However, they did not develop any practical tools or conduct detailed practical analysis.

Andrew Rathbun et al. (Rathbun et al., Ackerman) conducted reverse engineering of the Windows Diagnostics service binary and addressed the basic methods to analyze *EventTranscript.db* with analysis tools such as Diagnostic Data Viewer and Powershell script provided by Microsoft. Abhiram Kumar (Kumar) unveiled a tool to parse *EventTranscript.db* and extract information, but after parsing several events through a simple SQL query, the results were only shown in csv form.

Tyler Thomas et al. (2021) used two diagnostic telemetry events related to initial USB driver installation and compatibility check as USB-based attack detection indicators. In addition, they discovered that the information remained in the Windows 10 memory dump and developed *usbhunt*, a Volatiliy plugin that detects USB-based attacks in the Windows 10 memory dump.

As a result, so far, there have been no published papers or publicly available tools that have analyzed how users' activities are recorded in Windows Diagnostics.

3. User behavior analysis

3.1. Overview of the Eventtrancript.db

Eventtranscript.db is a SQLite3 database file consisting of eight tables (*categories, event categories, event_tags, events_persisted, producers, provider_groups, sqlite_sequence, tag_descriptions*). Among those tables, a table containing data about actual events is the *events_persisted* table, and the columns that are important in that table are shown in Table 1.

The *payload* column in the *events_persisted* table contains the actual event contents, and the contents of the column are in JSON format. The column consists of fields such as *os* and *protocol* containing device information, *time* and *loc* containing environmental information, and *data* containing substantial data of the corresponding event. Among them, the key fields among the fields commonly recorded in most events are shown in Table 2.

In addition to the fields organized in Table 2, unique data for each event is stored in a *data* field. We used this field to analyze the user's behavior. The specifications of the fields used in this paper are summarized and disclosed in the GitHub Repository in a *csv* format,[2] and the results of user behavior analysis using them are covered in other subsections of this section.

Table 2
Fields commonly included in the payload column of each event.

Fields	Description
name	The name of the event.
time	Time the event was recorded
ext.loggingBinary	The name of the process that logged the event
os.name	The name of the operating system
os.ver	Release version of the operating system
protocol.devMake	The manufacturer of the device
protocol.devmodel	The model name of the device
loc.tz	Timezone of the device being used (based on UTC)

3.2. Attaching and detaching USB devices

Windows Diagnostics contains behavioral records for attached USB devices. A series of events are generated when installing the device's driver at the time of initial connection of the device, and when attaching and detaching the device. By analyzing the events that occur in this manner, it is possible to accurately know the exact time when the user attaches and detaches the USB storage device to the PC. In addition, through such series of events detailed information about the USB device manufacturer, device name, serial number, and even the mounted drive can be found. This subsection introduces the results of analyzing the sequence of events that occur when performing two activities (attaching and detaching the device) related to USB devices.

3.2.1. Attaching USB devices

When attaching a USB device to a Windows PC, a series of events starting with *Microsoft.Windows.Storage.Classpnp.DeviceGuid Generated* and ending with *Microsoft.Windows.Storage. StorageService.SdCardStatus* occur in Windows Diagnostics (Fig. 1). This includes a series of processes in which Windows recognizes the attached device, reads information of the device, and mounts the device for availability. In case of the *Mount* events, different events occur in accordance with the filesystem of the attached device.

3.2.2. Detaching USB devices

When a USB device is detached, a *Microsoft.Windows.Storage.Classpnp.DeviceRemoved* event is generated by default. Thereafter, a *Microsoft.Windows.FileSystem.NTFS.Surprise Remove* or a *Microsoft.Windows.Storage.StorageService.Surprise Removal* event selectively occurs (Fig. 2). Such an event would convey that the USB device is immediately detached from the PC without software procedures. It is possible to know exactly which USB device is detached, since the *DevicedRemoved* event also records the unique information of the USB device including the serial number.

3.3. Web browser activity

Windows Diagnostics records user behavior logs related to Edge browser built into Windows by default. Behavioral information found in Diagnostics includes information that cannot be found in other browser-related artifacts, such as creating and closing tabs. This paper analyzed five major behaviors related to web browsers.

[2] https://github.com/L4wk3R/DiagAnalyzer/blob/main/events_descr iptions.csv.

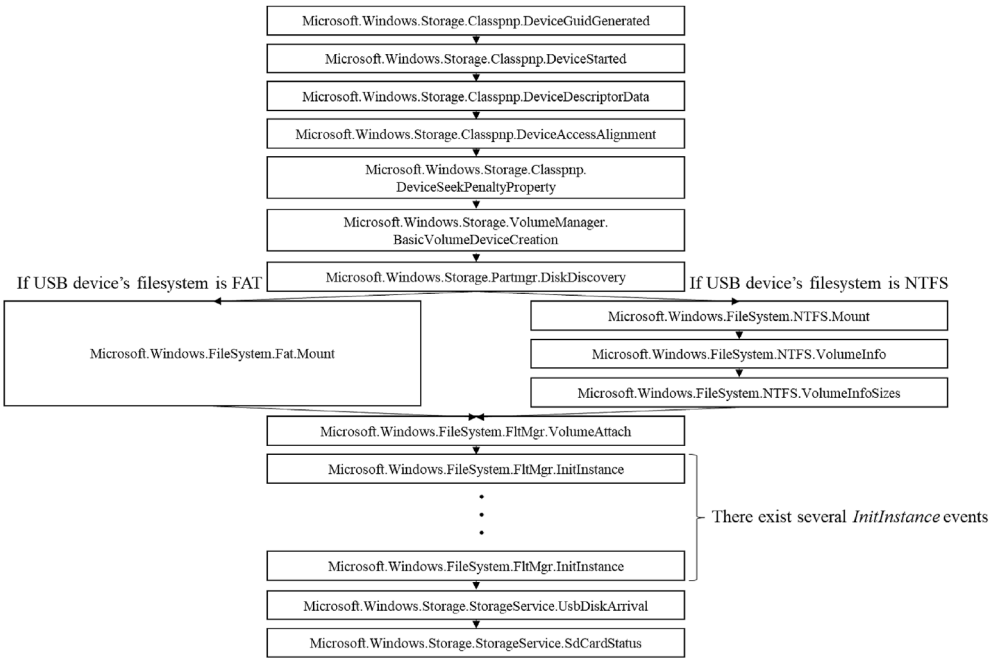

Fig. 1. A series of events that occur in Windows Diagnostics when the USB device is attached.

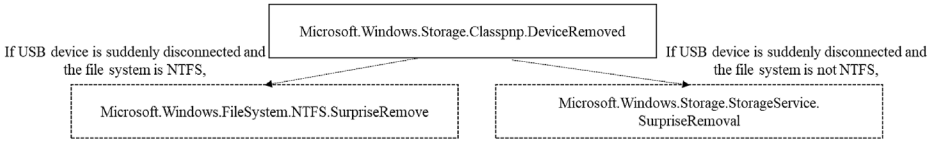

Fig. 2. A series of events that occur in Windows Diagnostics when the USB device is detached.

3.3.1. Web browser started and closed

When the Edge browser is executed, two events occur sequentially (Fig. 3). These events contain browser name, version information, and connection type information.

When the browser is terminated, an *Aria.<str>.Microsoft.WebBrowser.HistoryJournal.HJ_TabAllClosed* event occurs, and several *Aria.<str>.Microsoft.WebBrowser.HistoryJournal.HJ_TabClosed* events occur subsequently after the event. The number of *HJ_TabClosed* events is the same as the number of tabs that were active just before the browser was closed. In this case, *<str>* is a value different for each device with 16 hexadecimal values.

3.3.2. Tab created and closed

A single event occurs when a user creates or closes a tab in a web browser. When creating a tab, an *Aria.<str>.Microsoft.WebBrowser.HistoryJournal.HJ_TabCreated* event occurs, and when terminating the tab, an *Aria.<str>.Microsoft.WebBrowser.HistoryJournal.HJ_TabClosed* event occurs.

3.3.3. Visiting sites

In general, sites visited through web browsers can be found in *navigateUrl* field within *Aria.<str>.WebBrowser.HistoryJournal.HJ_BeforeNavigateExtended* events. However, if the *navigationUrl* field

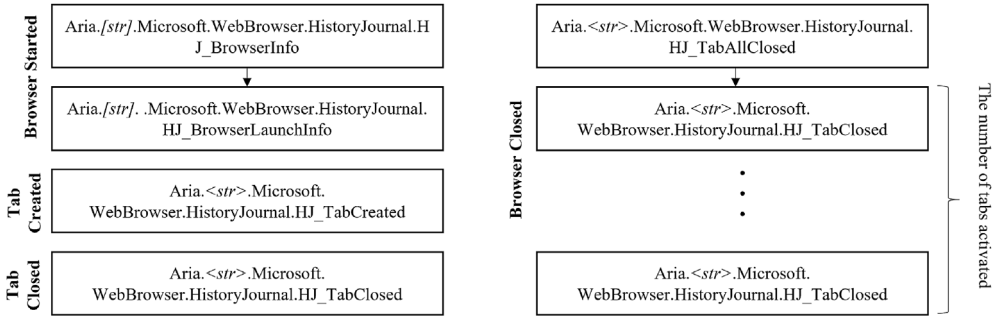

Fig. 3. A series of events that occur in Windows Diagnostics related to web browser activities.

within event is the only information collected, user's unvisited sites, such as communication with the ntp server, are also collected. In order to select and identify the sites the user actually visited, we repeatedly conducted several actions (search, click, url direct access) the user takes to visit a certain site and identified a series of commonly occurring events. As a result, four events occur sequentially in the case of sites visited by users themselves, and each associated event leads to the same *CorrelationGuid* (Fig. 4). In addition, the visit time, application name and version, connection type, url, and title information can be found in the payload of the corresponding events.

3.4. Wireless network activity

Windows Diagnostics records a series of events in the process of searching for a wireless network, connecting to the searched wireless network, and disconnecting. This subsection introduces the results of analyzing how Windows Diagnostics records a series of events in the process of scanning or attempting to connect to and disconnect from the WiFi AP.

3.4.1. Scanning WiFi AP

When a wireless LAN adapter connected to the device scans connectable WiFi APs, information on WiFi APs whose wireless signals are identified is recorded in the *WlanMSM.WirelessScanResults* event. The recorded information contains ssid and bssid information of the APs.

3.4.2. Trying to connect to WiFi AP

When a user attempts to connect to a WiFi AP, a *Microsoft.OneCore.NetworkingTriage.GetConnected.WiFiStartConnectionAction* event and a *Microsoft.OneCore.NetworkingTriage.GetConnected.WiFiConnectedEvent* event occurs sequentially regardless of whether the connection is successful or not, and these events can be connected through the same ssid and bssid. When connecting to the AP through clicking the UI, an additional *Microsoft.OneCore.NetworkingTriage.GetConnected.UXViewWiFiConnectClickedEvent* event occurs, and a *Microsoft.OneCore.NetworkingTriage.GetConnected.UXModelWiFiProcessUserInputAction* event occurs separately when the user directly inputs a password or ssid. Messages related to connection success and failure can be found in *wlanStatusCode* and *detailedStatusCode* within *Microsoft.OneCore.NetworkingTriage.GetConnected.WiFiConnectedEvent.*

3.4.3. Disconnecting from WiFi AP

When the us event and a *Wlanvc.WlanDisconnect* event occur. WiFi AP information, including ssid, bssid, authentication algorithm, and cipher algorithm, is also recorded at these events, along with disconnection reason information.

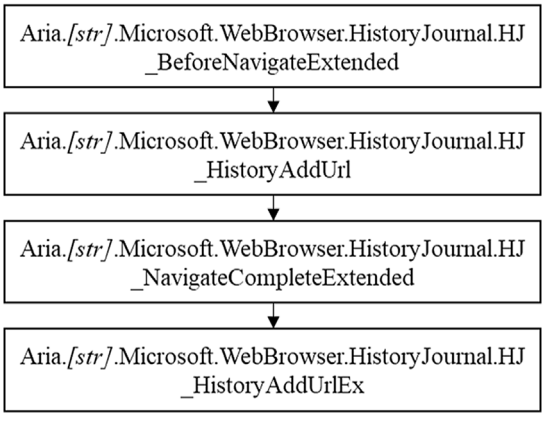

Fig. 4. A series of events that occur in common when a user visits a specific site with an Edge browser.

3.4.4. Tracking the user's location using bssid

When the wireless LAN card is activated, Windows continuously scans for nearby wireless APs that can be connected. The ssid and bssid of the scanned wireless AP are recorded in the *WlanMSM.WirelessScanResult* event. In addition, information of connected and disconnected wireless APs is also recorded in Diagnostics. There are already open services (Arkasha et al.; "Radiocells.org.") capable of searching for the location of a wireless AP opened through ssid and bssid (Fig. 5). These services manage the device and location information of the WiFi AP collected by themselves or through users as a database and provide services on a map. Accordingly, when a user scans or accesses an open wireless AP, the user's location information can also be found through wireless network connection information.

4. DiagAnalyzer

4.1. Design

Diaganalyzer is a python3 based script that receives *EventTranscript.db* extracted from the device as input and outputs the analysis results as a report in the form of html. *DiagAnalyzer* consists of the Analyzer and Report Generator (Fig. 6). Analyzer receives the *EventTranscript.db* file extracted from the filesystem and extracts device information, information about attaching and detaching USB devices, web browser behaviors information, and WiFi-related behaviors information. The Report Generator receives the analysis result from the Analyzer, visualizes the user's behavior, and then outputs an html-type report along with the extracted information.

4.2. Events creation

Event creation was conducted within VMWare Workstation Pro virtual machines running Windows 10 21H2 and Windows 11. In order to effectively collect and analyze diagnostic data, an optional data collection option was selected during the Windows installation process (this is the default option), and developer mode was activated. A complete apparatus of the system in this phase can be seen in Table 3. Three user behaviors were repeatedly performed in a configured environment to generate events, and then *EventTranscript.db* was extracted from the virtual machine. The extracted file was used to generate visualization result reports.

4.3. Visualization results

DiaAnalyzer visualizes the analysis results in tables and graphs. The information contained in the table is summarized in Table 4.

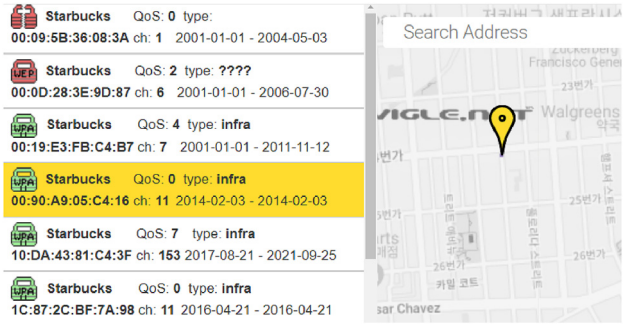

Fig. 5. Public service to find the physical location of WiFi AP using ssid and bssid (Arkasha et al.).

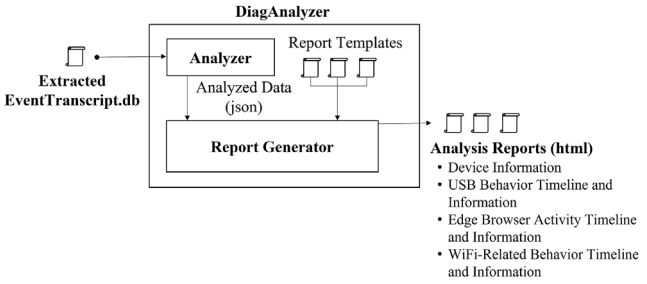

Fig. 6. Design of the *DiagAnalyzer*.

Table 3
Details of the events generation environment.

System	Summary
Processor	Intel Core i5-8250U
System type	64-bit
Virtual memory	8.00 GB
Operation System	Windows 10 21H2 Windows 11
Vmware version	Workstation Pro 15.1.0
Settings	**Summary**
Diagnostics & Feedback	Full
Developer mode	Enabled

Table 4
Information contained in reports generated by *DiagAnalyzer*

Category	Information
General	OS name OS version Device manufacturer Device model name Timezone
USB	Serial number Model name Manufacturer Filesystem
Web browser	Application Application version The time a web page was visited Title of the web page Url of the web page
Wireless AP	ssid bssid Hidden status Cipher algorithm Authentication algorithm Interface description

Fig. 7 shows the visualization results for each behavior from the information in the Analysis Report. In all graphs, the x-axis represents time. The upper graph in Fig. 7 shows an example of the USB behavior visualization result. The y-axis of the graph represents the name and manufacturer of the attached devices. Attaching and detaching events are respectively displayed in circles and triangles, and the time between attaching event and detaching event leads to a solid line. All symbols representing the event contain details related to attached USB devices such as device names, manufacturers, serial numbers, filesystems, and mounted volumes, which are activated by placing the mouse on the symbol.

The graph in the middle of Fig. 7 shows an example of the web browser behavior visualization result. The y-axis of the graph represents each representative behavior. The actions of a user opening a browser, opening a tab, visiting a site, closing a tab, and closing a browser are indicated by different symbols in each different coordinate. The circle that represents the event visiting a site contains detailed information including url, title, and connection type of the site visited, which are activated by placing the mouse on the circle.

The bottom graph in Fig. 7 shows the results of visualizing the behavior related to wireless network connection. The y-axis of the graph is the ssid of the WiFi AP found within the log. Events related to scanning, connection success, connection failure, and disconnection are respectively displayed in squares, circles, diamonds, and triangles. Each symbol contains information related to WiFi APs, such as ssid, bssid, hidden status, authentication, and cryptographic algorithms, and these are activated by placing the mouse on the symbol. Examples of reports containing Fig. 7 graphs can be found in *DiagAnalyzer*'s Github Repository.[3]

5. Discussion

DiagAnalyzer can be useful for digital forensics investigations for the following reasons.

First, the data leakage time using the storage medium can be accurately identified, as it is possible to know when the storage medium is attached and detached. That is, if the analysis results are properly combined with the shellbags and link files, they can be effectively used to investigate data breaches.

Second, it is possible to know the behavioral patterns in the Edge browser and find out how the user surfed the web before and after the incident. This may provide a possible relevance between the user and the incident. Investigators will be able to find significant clues to solving the case, if combined with other artifacts in the Edge browser.

Third, through the connected wireless network, it is possible to know the duration the user stayed in vicinity. This can be useful for tracking the user's alibi. It is particularly effective when users use WiFi in public places such as cafes or restaurants, and even if the organization uses a private network, this method can be used if the organization manages the asset list.

It is particularly useful in forensic audits in places such as companies that can centrally control user PCs. The reason is that options can be set at the time of distribution so that events can be fully generated, and even after distribution, options can be adjusted by utilizing management systems such as Patch Management System (PMS).

However, there are limitations in the analysis method of this paper. A typical limitation is that the function for specifically recording the user's behavior is not activated by default. Of course, the optional diagnostic data option is at the top of the list when setting diagnostic data during Windows's initial installation process. However, users can turn it off or change it to a low option at any time. Therefore, the effectiveness of the method is dependent on the environment. In addition, Windows Diagnostics does not record events for third-party web browsers such as Chrome, Firefox, or special-purpose web browsers (Tor Project; Bickford and Giura, 2016). Therefore, when these web browsers are used in the device for investigation, behavioral information cannot be extracted from Diagnostics, leaving no other option but to rely on other analysis methods (Rathod and Rathod; Marrington et al., 2012). Moreover, although the ssid and bssid of the wireless AP can be known from the Windows Diagnostics log, there are cases in which it is impossible to determine the physical location through this. Since the method presented in this paper relies on an open DB, it

[3] https://github.com/L4wk3R/DiagAnalyzer/tree/main/reports.