



Forensic implications of System Resource Usage Monitor (SRUM) data in Windows 8



Yogesh Khatri*

163 South Willard Street, P O Box 670, VT 05402-0670, USA

ARTICLE INFO

Article history:

Received 5 March 2014

Received in revised form 16 December 2014

Accepted 15 January 2015

Available online 11 February 2015

Keywords:

System Resource Usage Monitor

SRUM

Windows 8

Forensics

Process metrics

ABSTRACT

The Microsoft Windows 8 operating system has a newly added feature to track system resource usage, specifically process and network metrics over time. Process related information such as process owner, CPU cycles used, data bytes read/written, and network data (sent/received) are continuously recorded by a mechanism called System Resource Usage Monitor (SRUM). This paper describes the SRUM mechanism, its databases, Windows registry entries, data logging, and potential uses in a forensic examination. Prior to this applied research, no tools were available to parse the SRUM data to a usable format. As part of this paper, two scripts have been developed to aid forensic examiners who would want to read, parse, and decode this information from a forensic disk image.

© 2015 Elsevier Ltd. All rights reserved.

Introduction

System Resource Usage Monitor (SRUM) is a new technology that made its debut in Windows 8. SRUM tracks process and network statistics over time in a database. The information collected by SRUM includes process details, user details, CPU cycles, and network data sent or received by a particular process. Only a limited amount of this information is available to the end user – the bulk of the SRUM database is not displayed. Windows 8.1 which released in October 2013 continued to use this technology and expanded upon it.

SRUM offers forensic examiners an historical view into past system usage on a computer. From a forensic examiner's point of view, this database can be immensely useful in tracing user activity and linking process, user and network activity together. This paper examines SRUM databases, exposes the locations of these databases, the types of data stored, the formats and some working details of SRUM. A practical guide to extracting and decoding this

data is included in this paper. A limited analysis and interpretation of this data is presented to demonstrate its usefulness in a forensic investigation. This includes tracking processes in ways previously not possible, linking of process to network activity and tracking external or deleted processes. This new data has immediate applications in the incident response space as it can provide evidence of data copying, estimate the amount of ex-filtrated data from a computer at the same time tying it down to a specific user, process and time period.

Research methods

As of writing this paper, there are no existing works or published literature on SRUM. The SRUM feature is new in Windows 8 and, currently, there is no Microsoft documentation detailing the inner workings of SRUM. The research method used in this work for studying SRUM involved observation and experiment. Several experiments were conducted to study the SRUM mechanism as a black box. These are detailed in Section [Experiments](#) below.

Primarily two versions of Windows were studied, Windows 8 and Windows 8.1. Four test computers were

* Tel.: +1 626 344 9189.

E-mail address: yogesh@swiftforensics.com.

utilized which included a mix of laptops, desktops and virtual machines. On each test system, a baseline was taken, and then all test results were observed using forensic software, including EnCase and Autopsy. In addition, SRUM databases from several other personal computers were collected and analyzed to evaluate the consistency of the data structures observed during experimentation, and to validate the conclusions outlined here.

System Resource Usage Monitor

System Resource Usage Monitor (SRUM) is integrated into the new Diagnostic Policy Service (DPS). DPS enables problem detection, troubleshooting, and resolution for Windows components according to Microsoft ([Microsoft, Diagnostic Policy Service, 2009](#)). This service is enabled by default and configured to start automatically upon system startup on all Windows versions, including Enterprise versions. Internally, the system uses a number of extensions to monitor process, network, and energy resources. [Table 1](#) lists the extensions, associated DLL files and GUIDs that represent them. These GUIDs are the same on any Windows 8 or 8.1 systems.

'Windows Network Connectivity' (ncuprov.dll) has been added in Windows 8.1 and was not present earlier in Windows 8.

Data collection and update frequency

Process related data and network usage associated with processes are logged by SRUM. These data are collected for all desktop applications, system utilities, services, as well as Windows Store (Metro) Apps ([Microsoft, Meet Windows Store Apps](#)).

The process related information collected by SRUM includes full process path, process owner (user that launched the process), metrics for I/O data (bytes read & written in foreground/background), CPU cycle utilization (foreground/background), context switches, process network utilization (data uploaded/downloaded), and Windows Push Notifications (Notification type and data payload size). Network connectivity is also tracked, i.e. the start time and

amount of time the computer system was connected to a network. Windows also tracks the type of network, whether it is metered (3G, 4G ...) or non-metered (Ethernet or Wi-Fi) ([Microsoft, Metered Internet connections: FAQ](#)). On laptops and mobile devices, SRUM also collects system transition state and battery related information such as designated capacity, full charge capacity and available charge.

However, not all process or network related details are collected. Process details such as the command line arguments, DLL information, resource handles, thread information, or files accessed are not recorded by SRUM. Furthermore, network endpoint details like IP addresses, computer names, protocols or port numbers are not collected.

While SRUM continuously collects data on a running system, it only periodically updates the SRUM database. This update period is one hour by default. On any Windows 8 system, the default setting is to write out information at 30 min after every hour (example: 4:30, 5:30, 6:30 ...) unless a shutdown occurs, which triggers an immediate update of the SRUM.

Viewing SRUM data on windows

On a running Windows 8 system, SRUM data is viewable under Task Manager's 'App History' tab ([Gear, 2013](#)). However the view is very limited and most of the collected data are not shown, it is only an abridged summary of the SRUM database contents. For example, individual application runs are not displayed in the Task Manager view, only cumulative statistics are shown. In addition, application names shown in the Task Manager view are not those of the executable files, instead the 'File Description' strings from version information are used. Applications are tracked by their full path on disk, so two copies of the same application running from different locations will be shown separately. Application details for deleted executables and those that can no longer be found in their original location are grouped together under the name 'Uninstalled processes'. As most program installers extract and run from temporary locations such as %temp%, these also find themselves in this category as they are deleted from their temporary locations post install.

Details such as timestamps, application paths, and deleted application names are not on display. To obtain these details, it is necessary to read the data directly from the SRUM database and the Windows registry.

Retention

The SRUM service typically retains data for at least one month. This retention can be witnessed in Task Manager's App History tab which shows the starting date from which history has been displayed. However, much older entries are frequently found in the SRUM database. Users can also manually purge the collected data using the option 'Delete usage history' from the Task Manager as seen in [Fig. 1](#). However, this delete operation does not purge all information from the SRUM database. In experiments, even when the delete operation was selected, much of the data

Table 1
SRUM extensions, associated GUIDs and DLL files.

SRUM extension	GUID	DLL in System32
Windows Network Data Usage Monitor	{973F5D5C-1D90-4944-BE8E-24B94231A174}	nduprov.dll
Windows Push Notifications (WPN) SRUM Provider	{d10ca2fe-6fcf-4f6d-848e-b2e99266fa86}	wpnsruprov.dll
Application Resource Usage Provider	{d10ca2fe-6fcf-4f6d-848e-b2e99266fa89}	appsruprov.dll
Windows Network Connectivity Usage Monitor	{DD6636C4-8929-4683-974E-22C046A43763}	ncuprov.dll
Energy Usage Provider	{fee4e14f-02a9-4550-b5ce-5fa2da202e37}	energyprov.dll

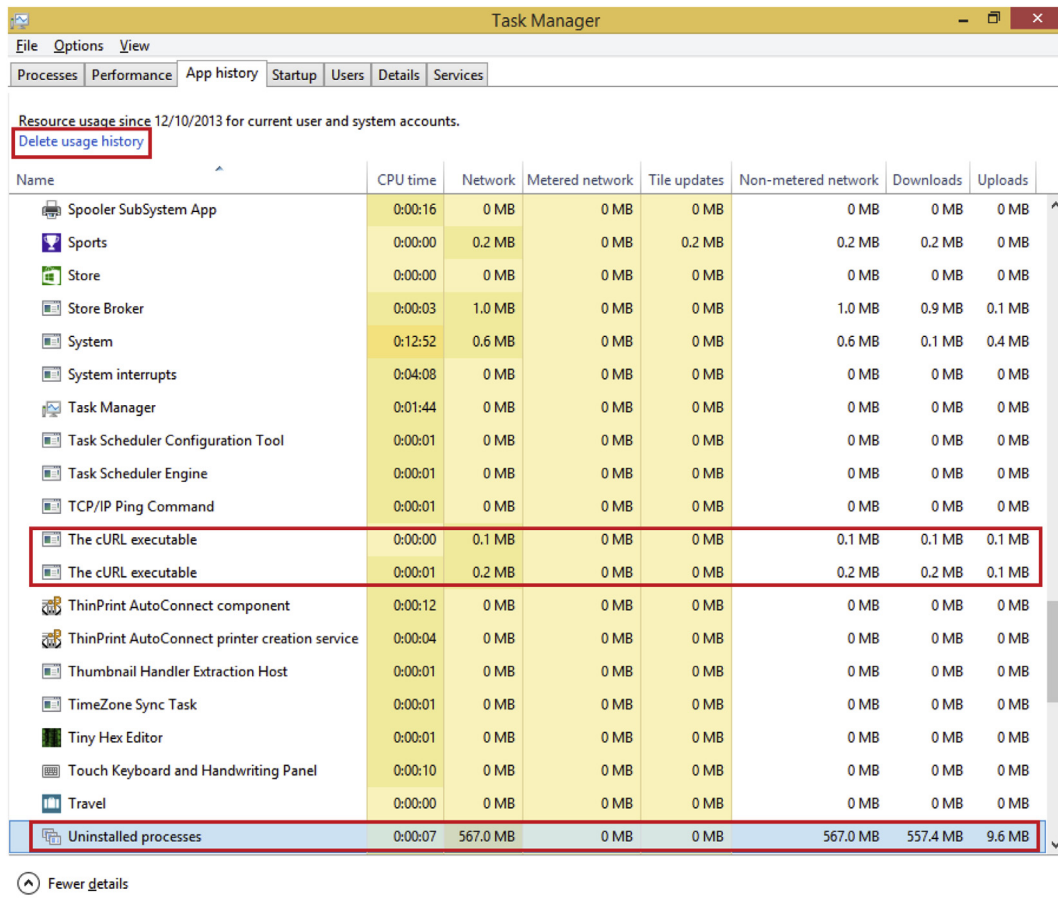


Fig. 1. Task manager screenshot showing App History. Here we can see two entries for 'The cURL executable', as it is being run under two different names, curl.exe and curl2.exe.

were still in the database even after several days and several shutdowns. However, the Task manager appears to obey the deletion instruction command, and only shows the newer data, even though older information still exists.

Technical details – location and format of SRUM data

SRUM data is stored in two locations on disk:

1. SRUDB.dat
2. Registry SOFTWARE hive

SRUDB.dat

The main SRUM database is located at path: %windir%\System32\sru\SRUDB.dat as shown in Fig. 2.

Name	Size	Type	Date Modified
\$B0	4	NTFS Index All...	1/31/2014 5:47:14 PM
SRU.chk	8	Regular File	1/31/2014 5:49:16 PM
SRU.log	64	Regular File	1/31/2014 5:49:16 PM
SRU00060.log	64	Regular File	1/31/2014 5:47:14 PM
SRUDB.dat	2,568	Regular File	1/31/2014 5:49:16 PM
SRUres00001.jrs	64	Regular File	10/31/2013 7:00:03 AM
SRUres00002.jrs	64	Regular File	10/31/2013 7:00:04 AM
SRUtmp.log	64	Regular File	1/23/2014 6:02:17 AM

Fig. 2. Folder 'C:\Windows\System32\sru' viewed in FTK imager.

Table 2

Tables contained in the SRUdb.dat file (Tables may not be in the same order on every computer system, hence table numbers may change).

Table name	Table number	Description
MSysObjects	0	Table metadata
MSysObjectsShadow	1	Table metadata
MSysObjids	2	Table metadata
MSysLocales	3	Table metadata
SruDbIdMapTable	4	Contains strings referenced in other tables
{DD6636C4-8929-4683-974E-22C046A43763}	5	Network Connectivity data
{D10CA2FE-6FCF-4F6D-848E-B2E99266FA89}	6	Application Resource usage data
{973F5D5C-1D90-4944-BE8E-24B94231A174}	7	Network usage data
{D10CA2FE-6FCF-4F6D-848E-B2E99266FA86}	8	Windows Push Notification data
{FEE4E14F-02A9-4550-B5CE-5FA2DA202E37}	9	Energy usage data
{FEE4E14F-02A9-4550-B5CE-5FA2DA202E37}LT	10	Energy usage data

This is an ESE (Microsoft, Extensible Storage Engine, 2012) database that Microsoft uses quite frequently on many of its products such as Exchange EDB files, Active Directory, Windows Mail, and Windows Search (Chivers and Hargreaves, 2011). The other files seen in this folder (in Fig. 2) accompanying SRUDB.dat are all part of the ESE infrastructure (Microsoft, Extensible Storage Engine Files, 2012) and are there to provide robust transaction logging and crash recovery support. Tools such as Nirsoft's ESEDatabaseView (Sofer, ESEDatabaseView, 2013) and Joachim Metz's Libesedb (Metz, 2012) are available to view and extract tables from an ESE database.

The database structure is simple consisting of 11 flat tables of which the first 4 describe table metadata such as table and column names, data types and locales. This is followed by 7 tables that hold SRUM information as shown in Table 2.

The table {DD6636C4-8929-4683-974E-22C046A43763} representing 'Network Connectivity Data' is only present on Windows 8.1.

Table 3

Column names, types, and description for the SruDbIdMapTable table.

Column name	Type	Description
IdType	Number (BYTE) that indicates data type in IdBlob column	Values (0 = Process Name and Path, 1 = Service Name, 2 = Windows App name, 3 = User SID)
IdIndex IdBlob	DWORD Unicode string for IdType (0,1,2) and binary SID (Microsoft, Security Identifiers) representation for IdType (3)	Primary key for this table

Table 4

Column names, types and description for table {DD6636C4-8929-4683-974E-22C046A43763}.

Column name	Type	Description
AutoInclId	WORD	Primary Key for table
TimeStamp	64 bit OLE timestamp	Date and time when entry was recorded by SRUM
ApplId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for Application/Service Name
UserId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for User SID
InterfaceLuid	QWORD	Locally unique identifier for interface
L2ProfileId	DWORD	
ConnectedTime	DWORD	Amount of time connected in seconds
ConnectStartTime	64 bit FILETIME timestamp	Timestamp when connection was established
L2ProfileFlags	DWORD	

Table and column descriptions

Only tables containing network and process information are detailed below.

SruDbIdMapTable. This table contains strings and Security Identifiers (SID) that are referenced in other tables. Information about its various columns is available below in Table 3.

{DD6636C4-8929-4683-974E-22C046A43763} (Windows Network Connectivity Usage Monitor). This table contains information about network connectivity, specifically identifying the start time and duration connected on a particular interface. The complete list of columns is available below as Table 4.

According to MSDN, the InterfaceLuid is a NET_LUID union (Microsoft, NET_LUID union) and internally consists of a structure split as 24 bits reserved, 24 bits of interface index and 16 bits of interface type. This gives information about the type of interface (Ethernet, Wi-Fi, 3G ...). The interface types are defined in the Windows SDK file 'ipifcons.h'.

{D10CA2FE-6FCF-4F6D-848E-B2E99266FA89} (Application Resource Usage Provider). This table stores resource usage statistics for applications that includes number of CPU cycles utilized, context switches, bytes read/written, and number of read/write operations among others. The complete list of columns is available below as Table 5.

{973F5D5C-1D90-4944-BE8E-24B94231A174} (Windows Network Data Usage Monitor). This table provides information about network usage. It provides specific statistics such as how much data was sent or received over a specific interface by a particular process. The complete list of columns is available below as Table 6.

{D10CA2FE-6FCF-4F6D-848E-B2E99266FA86} (WPN SRUM Provider). This table provides information about push

Table 5

Column names, types and description for table {D10CA2FE-6FCF-4F6D-848E-B2E99266FA89}.

Column Name	Type	Description
AutoInclId	DWORD	Primary Key for table
TimeStamp	64 bit OLE timestamp	Date and time when entry was recorded by SRUM
ApplId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for Application/Service Name
UserId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for User SID
ForegroundCycleTime	QWORD	Foreground CPU cycles consumed
BackgroundCycleTime	QWORD	Background CPU cycles consumed
FaceTime	QWORD	Unknown
ForegroundContextSwitches	DWORD	Foreground context switches
BackgroundContextSwitches	DWORD	Background context switches
ForegroundBytesRead	QWORD	I/O bytes read in foreground
ForegroundBytesWritten	QWORD	I/O bytes written in foreground
ForegroundNumReadOperations	DWORD	Foreground read operations
ForegroundNumWriteOperations	DWORD	Foreground write operations
ForegroundNumberOfFlushes	DWORD	Foreground flushes
BackgroundBytesRead	QWORD	I/O bytes read in background
BackgroundBytesWritten	QWORD	I/O bytes written in background
BackgroundNumReadOperations	DWORD	Background read operations
BackgroundNumWriteOperations	DWORD	Background write operations
BackgroundNumberOfFlushes	DWORD	Background flushes

notifications received by windows apps. The complete list of columns is available below as [Table 7](#).

SRUM in registry

In the registry, SRUM stores data in “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SRUM\Extensions” temporarily, until it is saved in the database.

The SRUM storage scheme in the registry is simple. Extension keys contain RecordSets which contain the individual records as seen in [Fig. 3](#). Each individual record holds the same data as in the SRUDB.dat database for the particular table that represents the same extension. The registry is a temporary holding place for this data which eventually gets moved to the database.

There exists a Key for every extension having its GUID as the key name. If the extension has data, then it will contain a single sub-key called ‘RecordSets’. The RecordSets key contains a DWORD Value named ‘RecordSetCount’ which represents the number of record sets contained therein. If this number is greater than zero, there will be sub-keys representing the record sets having names as ‘0’, ‘1’, ‘2’ and so on. A RecordSet holds records from a single collection session, so all timestamps in records within it will be the same or varying only by a few seconds.

Each of these RecordSet keys contains a DWORD Value named ‘RecordCount’ which shows the count of records contained within it. Keys for records are named as sequential numbers, the first one having the name ‘0’, then ‘1’, then ‘2’ and so forth. Each such record will contain the following values:

TimeStamp – 64 bit FILETIME value representing the date and time this entry was recorded.

ColCount – DWORD containing the count of columns.

UserId – Binary blob containing User SID.

ApplId – Binary blob containing application full path or service name. The format for this value is 4 bytes reserved (zero), next 4 bytes are string size in bytes followed by the data which is a Unicode string.

In addition to these values, there is the column data, which represents columns for that extension's table in the SRUM database. Binary values for columns are named as numbers starting with ‘0’ which represents column 4 for the corresponding table in the database (as columns 1–3 are already covered by values Timestamp, ApplId and UserId), the next value is ‘1’ representing column 5 and sequentially so on. All column data follow a simple format as shown in [Fig. 4](#) with the first 2 bytes representing column number, then 4 bytes reserved/unknown, followed by actual data. So, if the data for a particular column is a QWORD (8 bytes), the total size of the binary value would be $6 + 8 = 14$ bytes.

[Fig. 4](#) shows data for value ‘3’ representing column 7 (ConnectStartTime) for table {DD6636C4-8929-4683-974E-22C046A43763}, which contains Network Connectivity information.

Experiments

A number of experiments were conducted to simulate conditions such as IP theft (data exfiltration), evidence destruction, malware, and rogue process runs. The experiment setup and sequence of events are described here. Outcomes of these experiments are described in [Section Forensic Utility of SRUM information](#) (Forensic Utility of SRUM Information).

Data exfiltration

Two different methods of data exfiltration were tested, first using malware and next manually copying files over a network using Windows Explorer.

Table 6

Column names, types and description for table {973F5D5C-1D90-4944-BE8E-24B94231A174}.

Column name	Type	Description
AutoIncid	DWORD	Primary Key for table
TimeStamp	64 bit OLE timestamp	Date and time when entry was recorded by SRUM
Appld	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for Application/Service Name
UserId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for User SID
InterfaceLuid	QWORD	Locally unique identifier for interface
L2ProfileId	DWORD	
L2ProfileFlags	DWORD	
BytesSent	QWORD	Number of bytes sent by process
BytesRecv	QWORD	Number of bytes received by process

Malware

A system was infected with a malware sample 'cc8vrl.exe' to test if we can detect data exfiltration. The computer was then restarted and allowed to run for some time, and then periodically started up and shut down. As the trojan had registered itself to autostart, it would run every time windows was started. Table 8 below tracks the dates and times that the computer was on with the trojan running.

Data copying over SMB

An experiment was performed copying files over the network using Windows Networking (File and Network Sharing) by logging onto another computer over the same LAN using Windows login credentials for workgroup/domain and transferring files. In any corporate setup, this is a common mode of data transfer for copying files between systems or from file servers using Windows Explorer. These are also referred to as SMB file transfers. In this experiment, a couple of files totaling 2.1 GB in size were downloaded to the computer from a remote workstation. Table 9 below lists the timeline of operations on the computer.

Table 7

Column names, types and description for table {D10CA2FE-6FCF-4F6D-848E-B2E99266FA86}.

Column name	Type	Description
AutoIncid	DWORD	Primary Key for table
TimeStamp	64 bit OLE timestamp	Date and time when entry was recorded by SRUM
Appld	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for Application/Service Name
UserId	DWORD	Foreign key referencing SruDbIdMapTable.IdIndex for User SID
NotificationType	DWORD	Type of Notification (Raw, Badge, Tile = 128, Toast)
PayloadSize	DWORD	Notification data size in bytes
NetworkType	DWORD	Unknown (always zero)

Evidence destruction

Three specialist privacy protection tools, CCleaner (Piriform Ltd.), CleanAfterMe (Sofer, CleanAfterMe – Clean Registry entries and files in your system, 2010), and Privacy Eraser (Cybertron Software Co., Ltd.) were used to destroy forensic artifacts. Each tool was run selecting all options in an attempt to determine if SRUM created artifacts were affected. In addition, the tool SDelete (Russinovich, 2013) from Microsoft's Sysinternals Suite was used to wipe a few files.

Process runs

Several applications were run to determine if SRUM can provide any new information regarding process run dates, times, and process duration. A few of these were specifically monitored. The first of the monitored applications was Notepad++. Table 10 below lists the dates and times when Notepad++ was run.

In an attempt to determine process traces from applications that are no longer accessible from the system, two types of applications were run. First, a process was run from local disk and then the executable file itself was deleted. Next, applications were run from an external USB disk that was subsequently disconnected and never reconnected to the system. Table 11 below lists the dates and times for these.

Forensic Utility of SRUM information

Process metrics such as CPU cycles and ForegroundBytesRead may not be useful for addressing specific questions in a forensic examination, but this information could be useful to deduce behavior and usage pattern along with network information and timestamps. Analytics and metrics can be useful for in-depth analysis of malware, process, or user actions in question. As noted in Section Experiments above, experiments were conducted to simulate conditions such as IP theft, evidence destruction, malware data exfiltration, and rogue process runs. After each experiment, SRUM data was collected from the test systems and then analyzed in an effort to detect such activities and gauge the extent of damage using various metrics collected by SRUM. Other forensic artifacts such as Prefetch files were also used in these experiments to correlate information found in SRUM.

Using network information

Data exfiltration has been a very significant problem for organizations (Carvey, 2014). As stated by Carvey, unless you are monitoring the system and network at the time data were exfiltrated, it can be very difficult to determine what data were actually taken. SRUM allows us to get limited visibility into a computer's history by capturing system and network statistics over time.

Network usage information for a particular process or service can be useful to determine the extent of data transferred in or out of a particular computer in question. This could be used to determine how much data was

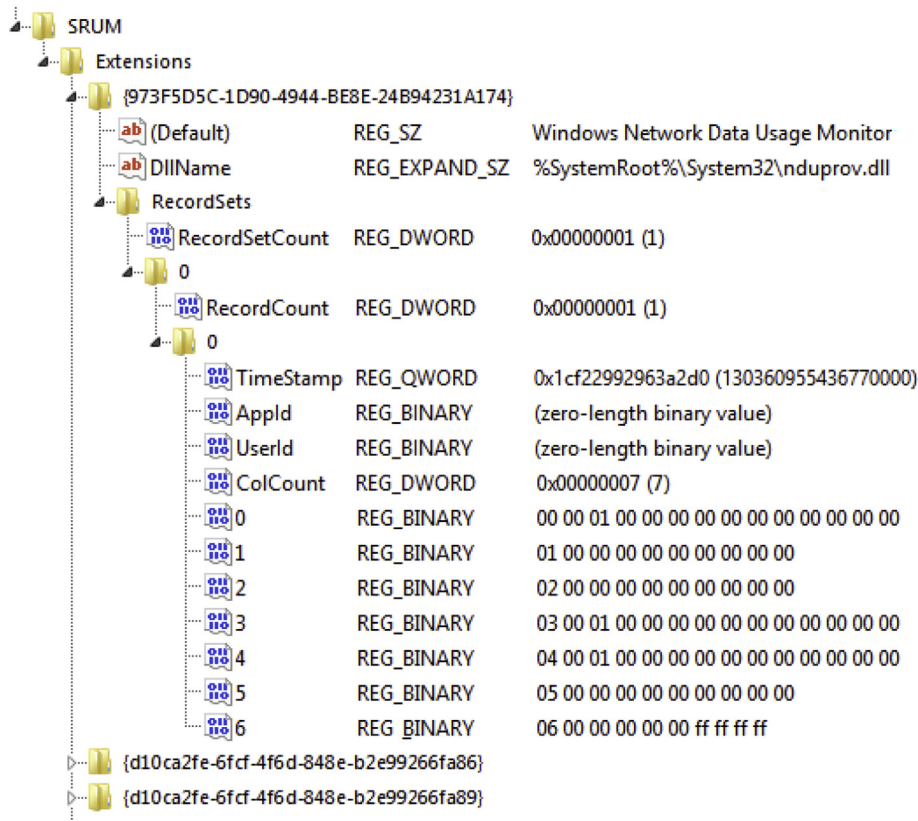


Fig. 3. SRUM data in the Windows registry.

transferred in browsing sessions, by viewing the network usage of browsers like Internet Explorer or Chrome. Data transferred to or from cloud storage can be measured by viewing the network usage of processes like Dropbox, iDrive, Google drive, etc. But more importantly, this information allows an investigator to determine how much data were transferred every hour as the SRUM collection interval is one hour as described in Section [Data collection and update frequency](#).

SRUM data available consists of raw sent/received statistics per process or service and they provide an approximation of how much data may be affected. While the statistics are exact, they may not represent the real or actual data size of files or data that were sent or received because it includes protocol overheads and possibly compressed data. Still, this information is very useful in data theft scenarios where large volumes of data are stolen.

Evidence of malware data exfiltration

Analyzing the output from the experiment described under Section [Malware](#), the amount of data sent out by a particular malware APT infection running under file name “ccvrl.exe” is shown in [Fig. 5](#). From the figure it can be seen that, in the interval between 21:04 to 21:30, about 28 MB (29,714,113 bytes) were sent and between 21:30 to 22:30 about 83 MB (87,050,023 bytes) were sent. In the next entry, the timestamp is not at 23:30; instead it is at 23:04. This is not a scheduled SRUM collection and would have occurred when the computer shut down. In our experiment, the computer was indeed shut down at 23:04 and started again around 12:15 on 1/1/2014. So the last entry represents data for time period 12:15 to 12:30. Similarly the first entry at 21:04 on 12/31/2014 was when the computer was shutdown. It was restarted at 21:20.

Table 8

Computer startup and shutdown times.

Date & time	Activity
12/31/2013 10:30	Computer started
12/31/2013 17:00	'ccvrl.exe' first started
12/31/2013 21:04	Computer shutdown
12/31/2013 21:20	Computer started
12/31/2013 23:04	Computer shutdown
1/1/2014 12:15	Computer started
1/1/2014 12:30	Computer shutdown

Name	Type	Data
		Reserved / Unknown
03	REG_BINARY	03 00 01 00 00 00 6d ac e6 b8 90 22 cf 01
		Column Number Column Data

Fig. 4. Format for column value in SRUM record.

Table 9

Timeline of data copying events.

Date & time	Activity
1/10/2014 13:17	Computer started
1/10/2014 14:10	File transfer initiated, download files totaling 2.1 GB in size from remote computer using Windows Explorer in a copy/paste operation
1/10/2014 14:14	File transfer completed
1/10/2014 18:43	Computer shutdown

SRUM collections are not triggered at system startup; hence these timestamps are not seen. By correlating this data with system startup times, we can further narrow the timeframes. So we revise our first interval where 28 MB was sent from 21:04–21:30 (26 min) to 21:20–21:30 (10 min) as the computer was switched off from 21:04 to 21:20.

Evidence of data copying over SMB

An SMB file transfer is performed via the windows explorer GUI using a copy & paste operation in the experiment from Section [Data copying over SMB](#). This will be recorded as network transfers initiated by the explorer process. By default, even with no user activity (just a connected live network), Windows Explorer will show a default pattern of usage with few KB sent and received every hour. This default pattern may vary depending on network type and usage of network drives. But in case of sudden transfer of large amount of data, it clearly stands out in the data. In [Fig. 6](#), 2.1 GB of files was downloaded by the workstation from an intranet site. By viewing these data, an investigator can easily conclude that 2.1 GB was downloaded between 13:30 and 14:30.

CPU time

CPU time is the amount of processor time that a particular application has used. This is not a measure of time for which the application is running. Most applications will utilize sleep when waiting for resources or user input; this will increase run time but not CPU time. SRUM does not log CPU time, instead CPU cycles are used which given the processor speed (cycles/second) can be roughly estimated as–

$$CPUtimeconsumed = \text{NumberofCyclesConsumed} / \text{CPUSpeed}$$

Table 10

Timeline of Notepad++ activity.

Date & time	Activity
1/15/2014 12:10	Computer started
1/15/2014 12:14	Notepad++ launched
1/15/2014 12:15	Notepad++ exit
1/15/2014 12:19	Notepad++ launched
1/15/2014 12:23	Notepad++ exit
1/15/2014 12:34	Notepad++ launched
1/15/2014 15:40	Notepad++ exit
1/15/2014 15:47	Computer shutdown

Table 11

Timeline of application activity.

Date & time	Activity
12/30/2013 22:33	Application 'curl' started from Local Disk
12/30/2013 22:35	Application 'curl' started from Local Disk
12/31/2013 21:00	Application 'curl' started from Local Disk
12/31/2013 21:16	Application 'curl' deleted
1/10/2014 15:09	USB disk connected
1/10/2014 15:10	Application 'viExploder' run from USB disk
1/10/2014 15:12	Application 'Diskview' run from USB disk
1/10/2014 15:24	USB disk disconnected

If unknown, CPU speed information can be obtained from the system's registry under 'HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor'. ([Microsoft, Different ways to determine CPU speed in Windows XP or in Windows Server 2003, 2008](#)).

For example, if your CPU speed happens to be 2.0 GHz (2000 MHz) which is 2,000,000,000 cycles/second and the cycles consumed is 24,000,000,000, then

$$CPUtime = 24,000,000,000 / 2,000,000,000 = 120 \text{ s} \\ = 2 \text{ min}$$

Estimating actual run time from CPU time

On certain non-interactive programs that do not wait on resources, the actual run time will be close to the CPU time consumed. This can be used by a forensic investigator to calculate approximately for how much time it was running. Sdelete ([Russovich, 2013](#)) is a simple, effective and popular file wiping tool from Microsoft (Sysinternals) which by default (no options used) wipes a file by doing a one pass write over the data. Sdelete was used to wipe a few files as part of the evidence destruction experiment noted in Section [Evidence destruction](#). The CPU time on a tool like sdelete will almost match the time that the process took to complete because on most computers, it will be consuming 100% of the CPU and the observed times reflect that as seen in [Fig. 7](#).

One caveat is that this kind of analysis is only valid for single function programs like sdelete, ideally for command line utilities that do not have any user interaction once started and do not wait on resources. Complex data destruction programs such as CCleaner will have many other I/O functions and attributing entire I/O to a single operation would always be wrong. Every application would have to be similarly tested before drawing conclusions.

Better process tracking

For a process run on Windows 8, its Prefetch file provides dates and times for the first time that the application was run ([Wade, 2010](#)) and also the last eight times ([Atkinson, 2013](#)) it was run. For frequently run applications or any application that has run more than 9 times, there is no available data in the Prefetch file for prior process runs. Data from SRUM may be able to fill in those gaps and determine if a process was indeed run during the interval unaccounted for by record in the Prefetch file. For example,

TimeStamp	AppId	InterfaceType	BytesSent	BytesRecvd
12/31/2013 21:04	\device\harddiskvolume2\users\test1\cc8vrl.exe	ETHERNET_CSMACD	343,413,499	165,327
12/31/2013 21:30	\device\harddiskvolume2\users\test1\cc8vrl.exe	ETHERNET_CSMACD	29,714,113	293,325
12/31/2013 22:30	\device\harddiskvolume2\users\test1\cc8vrl.exe	ETHERNET_CSMACD	87,050,023	2,329,832
12/31/2013 23:04	\device\harddiskvolume2\users\test1\cc8vrl.exe	ETHERNET_CSMACD	102,871,358	8,152,986
1/1/2014 12:30	\device\harddiskvolume2\users\test1\cc8vrl.exe	ETHERNET_CSMACD	4,561,958	32,341

Fig. 5. Evidence of trojan/malware process 'cc8vrl.exe' sending data out, seen in SRUM Application Resource Usage data.

on one test computer, there were 139 entries in SRUM for the executable Winword.exe representing several runs between 27th December 2013 and 21st February 2014. Prefetch files only stored the last eight dates and the oldest date from prefetch file was 19th February 2014. The gap between 27th December and 19th February could be filled in with data from SRUM. Precise startup times are not available from SRUM but it is possible to determine that the application was run on a particular date.

Another shortcoming of Prefetch files is that they do not record how long a process was running. Did it finish immediately, after an hour, or did it run during the entire logon session? This information can be discerned from SRUM data.

As noted in Section [Data collection and update frequency](#), when an entry for a process run exists in SRUM, then that process must definitely have been run within the last one hour (from that run timestamp) or that the process was launched earlier and was still running when the entry was made. Along with the precise process start timestamp from the Prefetch file, it can be determined whether a process was still running an hour later.

As an example, [Fig. 8](#) shows that Notepad++ was first run on 1/15/2014 at 12:14, then again at 12:19. A SRUM entry for Notepad++ was made at 12:30. This SRUM entry will contain resource usage information for both previous launches. As per Prefetch data, Notepad++ is run a third time at 12:34. There are no more executions until many days later. There are SRUM entries for Notepad++ at 13:30, 14:30, 15:30 and 15:47. This means that the third instance of Notepad++ continued to run from 12:34 till at least 15:31 which is approximately three hours. The end time is approximated as 15:31 because the last SRUM entry for

Notepad++ is at 15:47, and that means the program could have exited any time between 15:31 and 15:47.

Pin process run to a user

With every process or network metric, SRUM stores the corresponding user's SID, identifying the user account that launched the process as shown in [Fig. 9](#). This process-to-user mapping is very useful in a forensic examination. On a multi-user system, the exact user account that initiated the program can be identified. If rogue, disabled, or anomalous user accounts are seen in SRUM as owners of processes, this can help in identifying and investigating a security compromise ([Malin et al., 2012](#)). Without SRUM, process owner information is unavailable to a forensic examiner unless process tracking ([Smith, 2013](#)) was explicitly turned on and configured to save this information to event logs on the system.

Track Windows Store app runs

All desktop applications are associated with an executable file which is its initial launch point. The launch of every executable creates a trace as a Prefetch (.pf) file, thus making it easy to track program launches. However, in the case of Windows Store Apps, Prefetch files are not always created because many apps are not associated with an executable file. Furthermore, due to the nature of app runs, apps do not just transition from 'not running' to 'running' state and vice-versa. According to Microsoft, apps also have a 'suspended' state ([Microsoft, Application lifecycle](#) (Windows Runtime apps)), which becomes active as soon as it is no longer in the foreground, and an app can go from suspended state back to running state seamlessly. Hence,

TimeStamp	AppId	InterfaceType	BytesSent	BytesRecvd
1/10/2014 13:30	\Device\HarddiskVolume2\Windows\explorer.exe	ETHERNET_CSMACD	15,948	17,896
1/10/2014 14:30	\Device\HarddiskVolume2\Windows\explorer.exe	ETHERNET_CSMACD	355,242	2,271,147,907
1/10/2014 15:30	\Device\HarddiskVolume2\Windows\explorer.exe	ETHERNET_CSMACD	16,254	13,091
1/10/2014 16:30	\Device\HarddiskVolume2\Windows\explorer.exe	ETHERNET_CSMACD	5,041	11,129
1/10/2014 17:30	\Device\HarddiskVolume2\Windows\explorer.exe	ETHERNET_CSMACD	20,233	51,734

Fig. 6. Data copied/downloaded using Windows Networking shows up in SRUM Network Data Usage.

AppId	ForegroundCycleTime	Calculated CPU Time	Actual Run Time
\Device\HarddiskVolume2\temp\sdelete.exe	171,440,219,062	67.84337913	69 seconds
\Device\HarddiskVolume2\temp\sdelete.exe	15,552,074,647	6.154362741	7 seconds

Fig. 7. CPU Time matches actual run time for some applications, seen in SRUM Application Resource Usage data.

Prefetch App Launch Time	1/15/2014 12:14	
Prefetch App Launch Time	1/15/2014 12:19	
	1/15/2014 12:30	SRUM Data Collection
Prefetch App Launch Time	1/15/2014 12:34	
	1/15/2014 13:30	SRUM Data Collection
	1/15/2014 14:30	SRUM Data Collection
	1/15/2014 15:30	SRUM Data Collection
	1/15/2014 15:47	SRUM Data Collection
Prefetch App Launch Time	2/6/2014 17:50	
	2/6/2014 16:30	SRUM Data Collection

Fig. 8. Timeline view for Notepad++ application with data from prefetch and SRUM entries for the same application.

TimeStamp	AppId	UserId
12/13/2013 10:12	\Device\HarddiskVolume2\Windows\System32\cmd.exe	S-1-5-21-2003239035-2660515516-4832422-1001
12/13/2013 13:21	\Device\HarddiskVolume2\eroc_local.exe	S-1-5-21-2003239035-2660515516-4832422-1004
12/16/2013 20:00	\Device\HarddiskVolume2\Temp\scan.exe	S-1-5-21-2003239035-2660515516-4832422-1004

Fig. 9. Snippet of Application SRUM Provider data showing process to user mapping.

tracking Windows app runs becomes a challenge. However, resource usage information for such apps can still be found in SRUM data (see Fig. 10). If an app is using resources, then it is a definite indication of that app running.

Tracking external processes

Processes run from external media sources such as CDs or USB disks can also be seen in SRUM. Such processes can be easily recognized from their paths. By default, on a Windows installation, the 'C:\' drive is on the second partition; however this may vary due to additional OEM recovery partitions. All applications are referenced by their physical location. For example, the location for "explorer.exe" on the "C:\" drive appears as '\Device\HarddiskVolume2\Windows\explorer.exe' which references HarddiskVolume2. If the path references another volume such as 'HarddiskVolume3,' 'CdRom0,' or another value, then it is referring to a different partition quite possibly on an external device. Looking for such external device paths can reveal previously run suspicious executables. In Fig. 11, the Application Resource usage data shows traces of applications run which were run from a USB disk described as an experiment in Section Process runs, identified by physical location \Device\HarddiskVolume3.

Tracking deleted processes

An application or process that was previously run on the system and has since been deleted will leave traces in SRUM data. In experiments involving programs that were downloaded, executed, and subsequently deleted from disk, SRUM appeared to periodically remove older processes for which files no longer existed on disk from its database. The exact parameters or conditions for this removal of older SRUM records could not be determined, but it was noted that traces from some of the deleted applications survived beyond the default one month period typically retained by SRUM, while in some other cases they did not. Hence, while it is not guaranteed that traces from deleted applications may always be found here, it is still worthwhile to consider this artifact for this purpose.

As noted in Fig. 1, the Task Manager calls these 'Uninstalled processes' and does not display the name or path of such applications. Parsing SRUM data is the only way to view the missing information. Fig. 12 shows SRUM network data for an application program 'curl.exe' which was deleted from the system on December 31 at 21:16, as part of the experiment described in Section Process runs. The data shown here are from the SRUM database extracted on February 14, approximately a month and half after the application was deleted from disk. This entry and its

TimeStamp	AppId	ForegroundCycleTime	BackgroundCycleTime
2/5/2014 22:30	Microsoft.BingMaps_2.0.2530.2317_x86_8wekyb3d8bbwe	8,923,467,959	172,943
2/5/2014 23:09	Microsoft.BingFoodAndDrink_3.0.1.337_x86_8wekyb3d8bbwe	56,812,684,165	6,059
2/5/2014 23:09	Microsoft.BingWeather_3.0.1.203_x86_8wekyb3d8bbwe	68,081,046,029	412,063
2/5/2014 23:09	Microsoft.BingHealthAndFitness_3.0.1.335_x86_8wekyb3d8bbwe	66,544,707,234	552,986,680
2/6/2014 16:30	Microsoft.BingHealthAndFitness_3.0.1.335_x86_8wekyb3d8bbwe	-	7,565,487,943
2/6/2014 16:30	Microsoft.BingWeather_3.0.1.203_x86_8wekyb3d8bbwe	-	1,271,285,815
2/6/2014 16:30	Microsoft.BingFoodAndDrink_3.0.1.337_x86_8wekyb3d8bbwe	-	2,101,257,210

Fig. 10. Snippet of WPN SRUM Provider data showing Windows App Resource usage information.

TimeStamp	AppId
1/10/2014 15:30	\Device\HarddiskVolume3\ext\viExploder.exe
1/10/2014 15:30	\Device\HarddiskVolume3\Sysinternals\DiskView.exe

Fig. 11. Traces of programs run from external disks in SRUM Application Resource usage data.

counterpart in the application resource SRUM data has survived, but not all deleted processes traces survive that long.

Unknown and untouched by popular anti-forensic tools

In any forensic investigation, the use of anti-forensic or counter-forensic tools (Kessler, 2007) is always a concern. If they have been used on the computer in question, then valuable forensic artifacts may have been deleted or wiped. Specialist privacy protection tools like CCleaner, CleanAfterMe, or Privacy Eraser are notorious for deleting forensic artifacts such as Prefetch files, event logs, link files, program histories, and registry items. These three popular tools have been tested and reviewed to determine if they alter or delete any information from the SRUM databases. They currently do not touch SRUM data at all, nor do they claim to do so in their listed features. Features and documentation of several other anti-forensic tools were reviewed but no tools list clearing App History or SRUM as a feature.

Guide to decoding SRUM data

The following software tools are required to extract and decode SRUM data from a disk image:

1. Libesedb
2. Encase Forensic v6 (to run custom enscripts)
3. Microsoft Excel or OpenOffice Calc or similar wordprocessor

Extracting data from SRUDB.dat

To analyze SRUM data, the SRUM database files including the SRUDB.dat file need to be extracted from location 'C:\windows\system32\sru\'. Copy out the entire folder to a computer where libesedb is installed. Use libesedb's esedbexport tool to parse the database and output tab-delimited files, one for each table in the database. Fig. 13 shows libesedb parsing a single SRUDB.dat file.

These files can then be easily imported into any spreadsheet software like Microsoft Excel or OpenOffice Calc. However, before doing that, there are still many columns that need to be decoded and database foreign keys

that need to be resolved for easy reading. For this purpose, an Encase EnScript (Decode SRUM.enscript) has been developed that will read all the tab-delimited files (output from libesedb) and generate a new set of tab-delimited files with decoded dates and times, decoded SIDs, decoded Interface LUIDs, resolved AppIds and UserIds. This script can be requested by sending an email to the author. The resulting files from the EnScript output are now ready for viewing and analysis in any spreadsheet software.

Extracting data from registry SOFTWARE hive

SRUM data in the registry will contain the latest collected information that has not yet been moved to the database. It has been noted that information is populated into the registry only when the system is shutdown, otherwise all data is directly written to the SRUDB.dat database directly every hour. To analyze these data, they must be extracted from the registry. A second Encase EnScript (Decode SRUM from registry.enscript) has been developed for this purpose. This EnScript will read data in the registry directly from a loaded disk image in Encase and save the information into a set of tab-delimited files. The EnScript automates the process of finding the registry hive, mounting it, reading SRUM registry data and decoding timestamps and Interface LUIDs. This EnScript can also be requested by sending an email to the author. The resulting tab-delimited files from the script output can be easily imported into any spreadsheet software for viewing.

Observations and further research scope

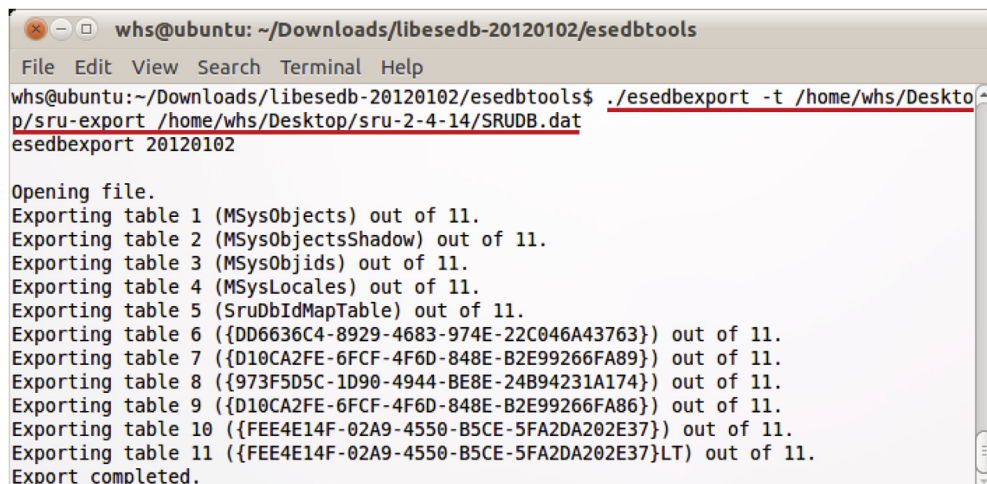
Processes that run quickly and die without using up significant resources of CPU time or disk I/O are sometimes not recorded in the SRUM database. The exact minimum threshold values or conditions for a process to definitively be recorded by SRUM have not been determined, but this is the general behavior noted.

The distinction between Foreground and Background is not quite clear at this point, i.e., what Windows classifies a read or write operation as foreground or background. In the context of a Windows app, background means a minimized app, running but no longer in focus on the main screen. It is unknown how this is applied to Windows desktop applications or services that have no user interface/interaction.

It is unclear yet whether I/O metrics represent only device, disk and control I/O, all of them, or any other I/O

TimeStamp	AppId	BytesSent	BytesRecv
12/30/2013 22:44	\device\harddiskvolume2\users\SID\downloads\curl.exe	2,648	166,546
12/31/2013 21:04	\device\harddiskvolume2\users\SID\downloads\curl.exe	3,434	165,327

Fig. 12. Deleted application traces in SRUM.



```
whs@ubuntu: ~/Downloads/libesedb-20120102/esedbtools
File Edit View Search Terminal Help
whs@ubuntu:~/Downloads/Libesedb-20120102/esedbtools$ ./esedbexport -t /home/whs/Desktop/
p/sru-export /home/whs/Desktop/sru-2-4-14/SRUDB.dat
esedbexport 20120102

Opening file.
Exporting table 1 (MSysObjects) out of 11.
Exporting table 2 (MSysObjectsShadow) out of 11.
Exporting table 3 (MSysObjids) out of 11.
Exporting table 4 (MSysLocales) out of 11.
Exporting table 5 (SruDbIdMapTable) out of 11.
Exporting table 6 ({DD6636C4-8929-4683-974E-22C046A43763}) out of 11.
Exporting table 7 ({D10CA2FE-6FCF-4F6D-848E-B2E99266FA89}) out of 11.
Exporting table 8 ({973F5D5C-1D90-4944-BE8E-24B94231A174}) out of 11.
Exporting table 9 ({D10CA2FE-6FCF-4F6D-848E-B2E99266FA86}) out of 11.
Exporting table 10 ({FEE4E14F-02A9-4550-B5CE-5FA2DA20E37}) out of 11.
Exporting table 11 ({FEE4E14F-02A9-4550-B5CE-5FA2DA20E37}LT) out of 11.
Export completed.
```

Fig. 13. Screenshot of libesedb usage – parsing SRUDB.dat with esedbexport.

types. One particular behavior noticed in this work is that data for write operations (ForegroundBytesWritten, BackgroundBytesWritten) are often twice or more in size than actual written data. One possible explanation could be that SRUM measures all data writes including that of kernel memory as all user mode writes result in data being written (copied) at least twice, once in user memory space, then in kernel space before being copied out to its destination.

Conclusion

Knowing how long a process was running, which user account launched it, and being able to narrow down data exfiltration down to an hour can be tremendously important information in a forensic investigation. SRUM data provides new information with an insight into process histories and network usage which can be useful in a detailed forensic investigation, especially incident response scenarios. By itself, these usage statistics do not give very valuable information but forensic examiners will benefit by combining and correlating several bits of information from varied sources such as Prefetch, Windows event logs, program caches, histories, and registry artifacts to reconstruct system events and user activities.

The historical information stored in SRUDB.dat will likely not provide very old data, as SRUM will periodically remove older entries. These artifacts will fade over time quickly, but by default a month's historical data should be available for analysis. Knowledge about SRUM is still limited, undocumented, and there does not appear to be any public API to access or delete SRUM data. However, it would not be very difficult to reverse engineer the 'sru-mapi.dll' file to figure out how to use the undocumented API; hence it is only a matter of time before anti-forensic tools catch up to this artifact.

Appendix A. Supplementary data

Supplementary data related to this article can be found at <http://dx.doi.org/10.1016/j.diin.2015.01.002>.

References

- Atkinson J. What's new in the prefetch for Windows 8. 2013, 9 21. Retrieved from: <http://www.invoke-ir.com/2013/09/whats-new-in-prefetch-for-windows-8.html>.
- Carvey H. Windows forensic analysis toolkit. 4th ed. Syngress; 2014.
- Chivers R, Hargreaves C. Forensic data recovery from the windows search. Digit Investig 2011;7(3–4):114–26.
- Gear G. Windows 8 task manager in-depth. 2013, 6 6. Retrieved 10 4, 2013, from Blogging Windows: <http://blogs.windows.com/windows/b/extremewindows/archive/2013/06/06/windows-8-task-manager-in-depth.aspx>.
- Kessler GC. Anti-forensics and the digital investigator. In: Proceedings of the 5th Australian Digital Forensics Conference. Perth: Edith Cowan University; 2007.
- Malin CH, Casey E, Aquilina JM. Malware forensics field guide for windows systems. Syngress; 2012.
- Metz J. libesedb. 2012. Retrieved 10 20, 2013, from: <https://code.google.com/p/libesedb/>.
- Microsoft. Different ways to determine CPU speed in Windows XP or in Windows Server 2003. 2008, 6 10. Retrieved from: <http://support.microsoft.com/kb/888282>.
- Microsoft. Diagnostic policy service. 2009, 1 8. Retrieved 10 1, 2014, from Technet: [http://technet.microsoft.com/en-us/library/dd393060\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/dd393060(v=ws.10).aspx).
- Microsoft. Extensible storage engine. 2012, 1 9. Retrieved from: [http://msdn.microsoft.com/en-us/library/office/gg269259\(v=exchg.10\).aspx](http://msdn.microsoft.com/en-us/library/office/gg269259(v=exchg.10).aspx).
- Microsoft. Extensible storage engine files. 2012, 1 9. Retrieved from MSDN: [http://msdn.microsoft.com/en-us/library/office/gg294069\(v=exchg.10\).aspx](http://msdn.microsoft.com/en-us/library/office/gg294069(v=exchg.10).aspx).
- Microsoft. Application lifecycle (Windows Runtime apps). Retrieved 10 4, 2013, from <http://msdn.microsoft.com/en-in/library/windows/apps/hh464925.aspx>.
- Microsoft. Meet Windows Store Apps. Retrieved 06 5, 2013, from <http://msdn.microsoft.com/en-in/windows/apps/hh974576.aspx>.
- Microsoft. Metered Internet connections: FAQ. Retrieved 10 04, 2013, from <http://windows.microsoft.com/en-us/windows-8/metered-internet-connections-frequently-asked-questions>.
- Microsoft. Security Identifiers. Retrieved 10 3, 2013, from MSDN: [http://msdn.microsoft.com/en-us/library/windows/desktop/aa379571\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa379571(v=vs.85).aspx).
- Microsoft. NET_LUID union. Retrieved 10 1, 2014, from MSDN: [http://msdn.microsoft.com/en-us/library/windows/desktop/aa366320\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa366320(v=vs.85).aspx).
- Piriform Ltd. CCleaner – PC Optimization and Cleaning. Retrieved 1 4, 2014, from <http://www.piriform.com/ccleaner>.
- Russinovich M. SDelete. 2013, 1 11. Retrieved 1 4, 2014, from Windows Sysinternals: <http://technet.microsoft.com/en-in/sysinternals/bb897443.aspx>.
- Smith RF. How to use process tracking events in the windows security log. 2013, 3 13. Retrieved from: <http://www.eventtracker.com/>

- [newsletters/how-to-use-process-tracking-events-in-the-windows-security-log/](#).
- Sofer N. CleanAfterMe – clean registry entries and files in your system. 2010. Retrieved 10 4, 2013, from: http://www.nirsoft.net/utills/clean_after_me.html.
- Sofer N. ESEDatabaseView. 2013, 6 19. Retrieved 1 4, 2014, from: http://www.nirsoft.net/utills/ese_database_view.html.
- Cybertron Software Co., Ltd. (n.d.). Privacy Eraser Pro. Retrieved 10 4, 2013, from <http://www.privacyeraser.com/>.
- Wade M. Decoding prefetch files for forensic purposes. 2010, 12 8. Retrieved from: <http://www.dfinews.com/articles/2010/12/decoding-prefetch-files-forensic-purposes-part-1>.