

TZWorks® USB Storage Parser (*usp*) Users Guide



Abstract

usp is a standalone, command-line tool used to extract USB artifacts from Windows operating system. The sources of the artifacts include the registry hives, setup API logs and event logs. It can analyze a live Windows machine or process discrete artifacts collected from another machine in an off-line manner

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v0.79 of ***usp***

Updated: Apr 15, 2024

Table of Contents

| | | |
|-------|--|----|
| 1 | Introduction | 2 |
| 2 | Which Windows artifacts are used | 3 |
| 2.1 | ReadyBoost Log..... | 5 |
| 2.2 | Event Logs | 5 |
| 2.3 | Other data that is available..... | 7 |
| 3 | Overview of the options | 7 |
| 3.1 | Processing USB Artifacts from a Live Windows System..... | 8 |
| 3.2 | Handling the primary and backup hives | 10 |
| 3.3 | Handling Volume Shadows from a Live System | 10 |
| 3.4 | Processing USB Artifacts from a 'dd' image of an NTFS disk | 11 |
| 3.5 | Processing USB Artifacts from a 'dd' image of an NTFS volume..... | 13 |
| 3.6 | Processing USB Artifacts off-line from extracted components | 14 |
| 3.6.1 | Specifying separate artifacts | 14 |
| 3.6.2 | Specifying a folder of artifacts | 15 |
| 3.7 | Processing USB Artifacts from an externally mounted drive..... | 16 |
| 3.8 | Pulling USB Artifacts from a Monolithic VMWare NTFS image. | 16 |
| 4 | Summary of all Options..... | 17 |
| 5 | Authentication and the License File..... | 20 |
| 5.1 | <i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner | 21 |
| 6 | Conclusions | 21 |
| 7 | References | 21 |

TZWorks® USB Storage Parser (*usp*) Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=13

Contact Information: info@tzworks.com

1 Introduction

usp is short for USB Storage (USBSTOR) Parser. It is a command line tool that can be scripted to work with other tools. It automates various manual techniques for extracting and analyzing different registry entries and Windows log files in order to pull together a report that documents the USB activity on a Windows computer. The report displays a summary of the USB device, timestamps of when the device was initially plugged, last time the device was plugged in, the serial number of the device, and various other metadata.

There are a number of use-cases that the Windows version of **usp** handles. For example, the tool can process USB artifacts from: (a) a live Windows system, ranging from Windows XP up to Win10, (b) an image of a Windows hard drive, (c) extracted registry hives and *setupapi* logs, (d) an external system drive that was mounted for analysis, and (e) a monolithic VMWare virtual disk file.

usp has been built so that it relies only on the standard operating system libraries. This means it does not require any extra libraries (DLLs) to be installed on the system for it to run. For any critical parsing, **usp** uses its own internal algorithms. For registry reading and traversals, it doesn't make use of the Windows API calls. Therefore, if the system you are analyzing has been compromised, **usp** should be able to extract what it needs and process the results without losing data. Since there is no installer for **usp**, it is easy to run directly from a USB stick or other portable device.

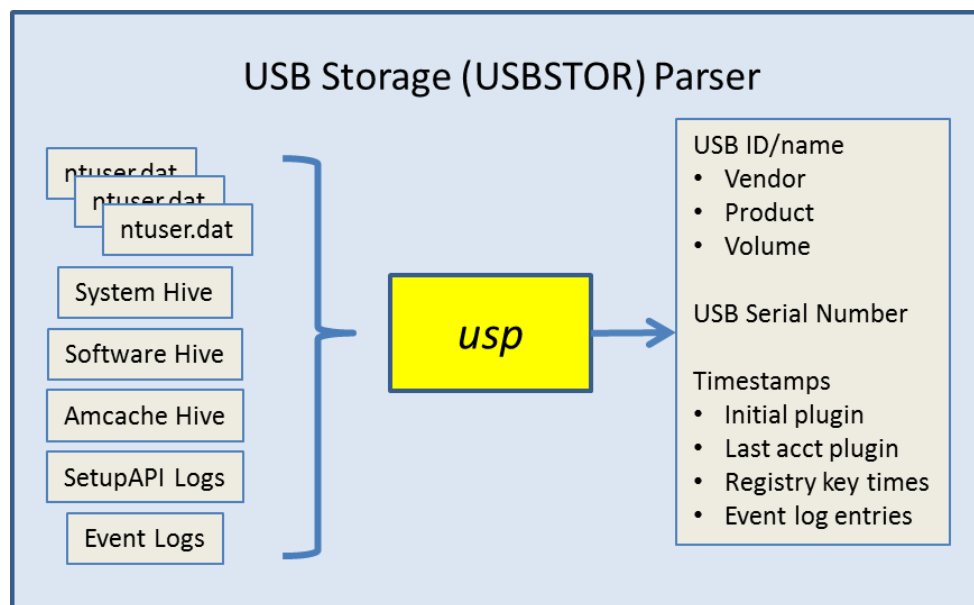
While **usp** gathers USB device statistics on Windows operating systems, it can be run on other operating systems in a limited mode. If one wishes to analyze Windows forensic artifacts off-line on Linux or Mac OS-X, there is a compiled version of **usp** to handle these operating systems as well.

usp can be downloaded from http://www.tzworks.com/prototype_page.php?proto_id=13. See the licensing agreement on the website for more details.

2 Which Windows artifacts are used

There are currently five different sources of Windows artifacts that can be used for **usp** to completely process USB device statistics. These include: (a) the *setupAPI* log(s), (b) the *system* hive, (c) the *software* hive, (d) the *user* registry hives, (e) the *AmCache* hive and (f) certain event logs.

The *setupAPI* log can be one or more files that identifies, amongst other things, when a USB device was initially plugged in. The *system* hive identifies which USB devices were registered with the Windows plug and play manager. Windows makes use of a number of registry keys to allow it to identify that same device quickly the next time it is plugged in. The *software* hive provides some additional information for those USB devices identified by the system hive. The *user* hives are used to associate which user account was logged on when the USB device was plugged in. This artifact can help identify when a user last plugged in the device. With the advent of the Windows 10 *Creators Update*, the *amcache* now has additional information when a device was registered, via the *InventoryDevicePnp* subkey. Finally, various event logs are also examined for USB artifacts and included in the results. As more research in Windows USB forensics becomes available, it can easily be incorporated into **usp** to enhance its reporting due to the extensible nature of its architecture.



When mapping the output to these artifacts, it can be confusing. Therefore, the following graphic shows which artifacts are represented in each of the two main output formats: (a) unstructured (long/verbose) output and (b) CSV output:

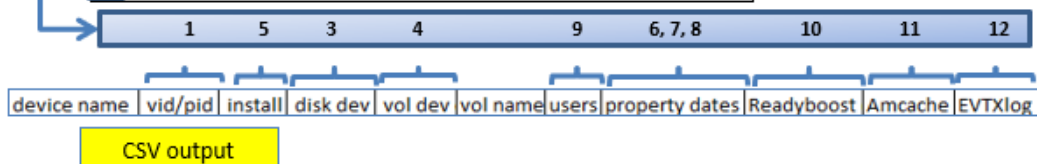
Mapping of timestamps extracted from *usp* and where they are outputted

1. Pulls date from subkeys that are 2 levels deeper in "USB" key
"HKLM\SYSTEM\CurrentControlSet\Enum\USB**\Service":
2. Pulls date from the subkeys that are 2 levels deeper than "USBSTOR"
"HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR**\":
3. Pulls date from the subkeys of parent key {53f56307-b6bf-11d0-94f2-00a0c91efb8b};
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceClasses\{GUID for disk device interfaces}"
4. Pulls date from the subkeys of parent key {53f5630d-b6bf-11d0-94f2-00a0c91efb8b};
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceClasses\{GUID for volume device interfaces}"
5. Pulls date from SetupAPI log
6. Pulls explicit date in value from the Property ID 0x0064 in "USBSTOR" [Equates to InstallDate]
"HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR**\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0064\":
7. Pulls explicit date in value from the Property ID 0x0066 in "USBSTOR" [Equates to LastArrivalDate]
"HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR**\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0066\":
8. Pulls explicit date in value from the Property ID 0x0067 in "USBSTOR" [Equates to LastRemovalDate]
"HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR**\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0067\":
9. Pulls date from the subkeys of parent shown, using the appropriate ntuser.dat hive
"NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2"
10. Pulls date(s) and maps volume serial numbers to devices, using the Software hive
"HKLM\Software\Microsoft\Windows NT\CurrentVersion\EMDMgmt**\":
11. Pulls date(s) from AmCache hive [InventoryDevicePnp subkey]
12. Pulls date(s) and event specific data for USB devices
(DF) Microsoft-Windows-Drivers-UserMode\4Operational.evtx [IDs 2010, 2102 – other event IDs are redundant]
(KP) Microsoft-Windows-Kernel-PnP\4Configuration.evtx [IDs 410, 420..]
(PD) Microsoft-Windows-Partition\4Diagnostic.evtx [ID 1006]
(SY) System.evtx [IDs 20001, 20002, 20003]

```

Device name: Kingston DataTraveler G3 USB Device
1 vid/pid key update [UTC]: 01/23/2018 23:22:18.329
2 ven/prod/rev key update [UTC]: 01/23/2018 23:22:18.329
3 Disk Device update [UTC]: 01/23/2018 23:22:18.331
4 Volume Device update [UTC]: 01/23/2018 23:22:18.343
5 SetupAPI Log dates: [Local] 01/13/2018 16:22:32.785
12 Eventlog dates: [UTC] EVTX_INS_DF#967826: 01/23/2018 23:22:20.650
    EVTX_REM_DF#967868: 01/23/2018 23:23:35.636
6 DEVPKEY InstallDate [UTC]: 01/13/2018 21:22:32.643
7 DEVPKEY LastArrivalDate [UTC]: 01/23/2018 23:22:18.234
8 DEVPKEY LastRemovalData [UTC]: 01/23/2018 23:23:35.356
Instance ID/Serial #: 0019e001089ff98015ad093180
...
9 Acct that mounted vol: tzlabs acct, on 01/23/2018 23:22:20.608 [UTC]
10 Vol serial/name/last time(s): f552-1ef2/test : 01/23/2018 23:22:18.521 [UTC]
11 InventoryDevicePnp [UTC]: 01/13/2018 21:23:45.134
    
```

Unstructured
(long) output



As one can see there are various sources for timestamp data, and some of them are redundant. The first source, which is common across all versions of the Windows operating systems, are the registry last modification times for their respective subkey path. One can also use the *SetupAPI* log(s) to extract installation time. In Windows 7, the device installation date property identifier should be present as well as the *EMDMgmt* timestamp(s). In Windows 8, the device last arrival/removal dates property identifiers may also be present. While it may seem redundant to display similar 'event type' timestamps, the extra data allows the investigator to corroborate when certain actions took place, and thus, increase the confidence the behavior suggested by the data was not influenced by anti-forensics techniques. Conversely, if there are inconsistent timestamps, then it tells the investigator that a closer look is warranted.

2.1 ReadyBoost Log

In the figure above, item 10 is the *ReadyBoost* artifact. This data initially became available in Vista. As background, whenever a new drive is connected to a windows box, the operating system will test that drive's read and write speed by creating a file on that drive and then deleting it. This result is logged in the *ReadyBoost* log. The timestamp associated with this entry was the time of the last test performed. In addition, the name of the disk is present and in some cases the size of the disk.

2.2 Event Logs

Added with the version 0.55 of *usp* is the ability to parse event log data from **evtx** type logs and integrate it with the USB results collected from the registry hives and setupapi logs. Since this is a new feature, it is still experimental in nature. Currently, up to four log files are can be analyzed: (a) *System.evtx*, (b) *Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx*, (c) *Microsoft-Windows-Kernel-PnP%4Configuration.evtx*, and (d) *Microsoft-Windows-Partition%4Diagnostic.evtx*.

From the logs listed above, certain events are enumerated and categorized into 3 areas: (a) when the USB device was inserted, (b) when the USB device was removed, and (c) when the USB device driver/service was deleted. Additional data can be extracted with the *Microsoft-Windows-Partition%4Diagnostic.evtx* log, such as the, partition table and volume boot record of the USB device (if present).

Since event log data can be noisy, in the sense, that many events can be recorded during one of the categories above (insert, remove or delete), the **usp** tool will collate clusters of events within a set interval and report each cluster as a significant event. Even though this clustering is done for reporting purposes, the tool provides traceability down to the record numbers used during a cluster operation, so the analyst can go back to a particular event log and look up the specific event record, if desired.

The following shorthand notation is used when reporting event log artifacts:

- a. One of the following prefixes:
 - INS (Insert USB device)
 - REM (Remove USB device),
 - DEL (Delete USB device).
- b. The above prefix is then followed by an underscore, followed by a two letter code:
 - DF (DriverFrameworks = *Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx*),
 - KP (Kernel PnP = *Microsoft-Windows-Kernel-PnP%4Configuration.evtx*),
 - PD (Partition Diags = *Microsoft-Windows-Partition%4Diagnostic.evtx*),
 - SY (*System.evtx*)
- c. Finally the record number is annotated.

Below is some output from two reports using a couple of different event logs.

The first example uses the **-csv/2t** format. In this case, **usp** breaks out each of the categorized events (insert, remove, and delete) into clusters and creates a separate row entry for each one. The overall

output merges the USB event log data with the registry hive and setupapi data. The traceability of which event record goes with the csv line output is annotated in the 'extra' column. The event log data in the example uses the syntax:

INS_KP#423, for *insertion* of device taken from *record# 423* in the *Kernel Pnp* (*Microsoft-Windows-Kernel-PnP%4Configuration.evtx*) event log,

REM_PD#286813, for *removal* of device taken from *record #286813* in the *Partition Diags* (*Microsoft-Windows-Partition%4Diagnostic.evtx*) event log

Etc.

| cmdline: <filelist> ... usp64 -pipe -csvl2t -pair_datetime -csv_separator " " -base10 [implied cmd: -event_res 2] | | | | | | EVTX data | |
|--|------|----------|--|-------------------------------|--------|---|--|
| date/time | MACB | source | type | short | dev | extra | |
| 10/16/2017 16:04:04.337 | ..C. | EVTX | Inserted (evtx: KernelPnp) | Generic Flash Disk USB Device | vid... | Clustered records: INS_KP#423; INS_KP#425; INS_KP#427 | |
| 10/16/2017 16:04:04.340 | ...B | REG | Installed (system: VenProdDeviceProdK) | Generic Flash Disk USB Device | vid... | | |
| 10/16/2017 16:04:04.375 | ..C. | REG | Inserted After Reboot (system: VolDevic | Generic Flash Disk USB Device | vid... | | |
| 01/18/2018 03:10:45.954 | ..C. | SETUPAPI | Scheduled Uninstalled (setupapi) | Generic Flash Disk USB Device | vid... | | |
| 01/18/2018 03:11:43.654 | ..C. | EVTX | Inserted (evtx: DriverFrameworks) | Generic Flash Disk USB Device | vid... | Clustered records: INS_DF#2; INS_PD#284812 | |
| 01/18/2018 03:15:46.769 | ..C. | EVTX | Removed (evtx: PartitionDiags) | Generic Flash Disk USB Device | vid... | Clustered records: REM_PD#284813; REM_PD#28481 | |
| 01/18/2018 04:05:42.578 | M.C. | REG | Last Insert (system: Vid/Pid time); LastAr | Generic Flash Disk USB Device | vid... | | |
| 01/18/2018 04:05:42.603 | ..C. | EVTX | Inserted (evtx: DriverFrameworks) | Generic Flash Disk USB Device | vid... | Clustered records: INS_DF#114; INS_PD#286200 | |
| 01/18/2018 04:05:43.080 | .A.. | REG | User Acct Mounted (ntuser: MountedDev | Generic Flash Disk USB Device | vid... | | |
| 01/18/2018 04:06:47.168 | ..C. | EVTX | Inserted (evtx: PartitionDiags) | Generic Flash Disk USB Device | vid... | Clustered records: INS_PD#286201; INS_PD#286202 | |
| 01/18/2018 04:06:47.186 | ..C. | EVTX | Removed (evtx: DriverFrameworks) | Generic Flash Disk USB Device | vid... | Clustered records: REM_DF#146; REM_PD#286203 | |
| 01/18/2018 04:06:53.176 | ..C. | REG | LastRemoval (system: DEVPKEY LastRem | Generic Flash Disk USB Device | vid... | | |

To see more detail about the device, and only the cluster event log timestamps, one can use the -v (or verbose) option. This option shows more clearly the other event log data when it comes to the partition table and volume boot record for the device, if it was available. This data shows the volume offset, number of bytes in the volume, disk signature, volume signature, and other disk/volume related data.

| | |
|---|---|
| usp - full ver: 0.55; Copyright (c) TZworks LLC License #1d38e7cc14a3070 is authenticated for business use and registered to TZworks LLC run time: 02/14/2018 05:39:13 [UTC]; Host: TZLABS-PC004; MachineInfo: windows;6.1.workstation.sp.1.0;6.1.7601;64 "cmdline: <filelist> ... usp64 -pipe -v -pair_datetime -base10" [implied cmd: -event_res 2] | |
| USB Device: 0 | Generic Flash Disk USB Device [Mass Storage] [TZWORKS] |
| Device name: | disk [volume] |
| Service: | 10/18/2018 04:05:42.578 |
| vid/pid key update [UTC]: | 10/16/2017 16:04:04.341 |
| ven/prod/rev key update [UTC]: | 10/16/2017 16:04:04.349 |
| Disk Device update [UTC]: | 10/16/2017 16:04:04.375 |
| Volume Device update [UTC]: | 01/17/2018 22:10:45.954 |
| SetupAPI Log dates: [Local]: | |
| Eventlog dates: [UTC]: | [EVTX_INS_KP#423: 10/16/2017 16:04:04.337 UTC] [EVTX_INS_PD#284812: 01/18/2018 03:11:43.654 UTC] [EVTX_REM_PD#284813: 01/18/2018 03:15:46.769 UTC] [EVTX_INS_PD#286200: 01/18/2018 04:05:42.603 UTC] [EVTX_INS_PD#286201: 01/18/2018 04:06:47.168 UTC] [EVTX_REM_PD#286203: 01/18/2018 04:06:47.186 UTC] |
| DEVPKEY InstallDate [UTC]: | 10/16/2017 16:04:04.340 |
| DEVPKEY LastArrivalDate [UTC]: | 01/18/2018 04:05:42.578 [other dates available: only showing the latest one] |
| DEVPKEY LastRemovalDate [UTC]: | 01/18/2018 04:06:53.176 [other dates available: only showing the latest one] |
| Instance ID/Serial #: | 8680c0e7&0 |
| Container ID: | 155c6c58-4fdd-59d0-9d5c-52f999151a3c |
| Volume ID: | 5dad7834-a7ef-11e7-8aab-000c29731513 |
| Disk ID: | 5dad7832-a7ef-11e7-8aab-000c29731513 |
| Volume name: | TZWORKS [E:] |
| Volume name update [UTC]: | 10/16/2017 16:04:06.517 |
| Parent ID Prefix: | <none found> |
| Vendor ID: | 058f |
| Product ID: | 6387 |
| Revision: | 8.07 |
| Vendor/product: | generic/flash_disk |
| USB hub/port used: | 3/ 6 |
| Acct that mounted vol: | tzlabs_acct on 01/18/2018 04:05:43.080 [UTC] |
| Partition table: | vol1;byte_offset:4980736;num_bytes:8173912064;disk_sig:2e47-c29a;vol_type:fat32 |
| Volume boot record data: | sysid:fat32;sector_size(bytes):512;cluster_size(bytes):4096;media:hard_disk;part |

2.3 Other data that is available

Aside from the timestamp data, **usp** displays other metadata about the USB device. Below is an example of this other data, with the timestamp data removed, to focus on the other output.

```
Device name: Kingston DataTraveler G3 USB Device
...
Instance ID/Serial #: 001cc0ec2f41c09145c20726&0
Driver: {4d36e967-e325-11ce-bfc1-08002be10318}\0008
Volume ID: 4eef2083-1af1-11e3-be77-24fd52334df7
Disk ID: 4eef2080-1af1-11e3-be77-24fd52334df7
Volume name: XFER
Parent ID Prefix: <none found>
Vendor ID: 0930
Product ID: 6544
Revision: 1.00
Vendor/product: kingston/datatraveler_g3
USB hub/port used: 03/01
...
```

The instance ID (or serial number), is one of the main pieces of data that links many of the various artifacts together. The volume identifier links the data in the system hive to the USB data in the user hive to correlate which user account mounted the USB device. The parent prefix identifier is more useful in the pre-Vista operating systems to provide linkages between data. The vendor ID, product ID, revision and product name are pulled directly from the registry information. Finally, the USB hub/port combination is extracted to record where the device was plugged into.

3 Overview of the options

There are a number of use-cases that **usp** was designed for. Below is a breakout listing which binaries are compatible with each use-case:

- Live Windows processing (Win32/64 binaries)
- Off-line processing of a 'dd' image of a disk (Win32/64, Linux32/64 and Mac OS-X 32/64 binaries)
- Off-line processing of extracted registry hives and setupAPI logs (Win32/64, Linux32/64 and Mac OS-X 32/64 binaries)
- Processing an external mounted drive (Win32/64 binaries)
- Processing a monolithic VMWare NTFS formatted virtual disk (Win32/64, Linux32/64 and Mac OS-X 32/64 binaries)

The various options above can be seen from the menu below. The rest of this paper discusses each of the use-cases in more depth and is diagramed with examples.


```
Administrator: Windows PowerShell

Usage

usp -sys <hive> -user <hive> -setupapi <file> [-sw <hive>] [-evtx <log>]
usp -livesys                               = pull from live computer
usp -image <file> [-offset <vol offset>]    = use image as source
usp -vmdk "file1 | file2 | .."              = process a VMWare volume
usp -drivenum <#> [-offset <vol>]           = use drive num as source
usp -partition <drive letter>               = use partition as source
usp -vss <index>                           = *** use vol shadow as source

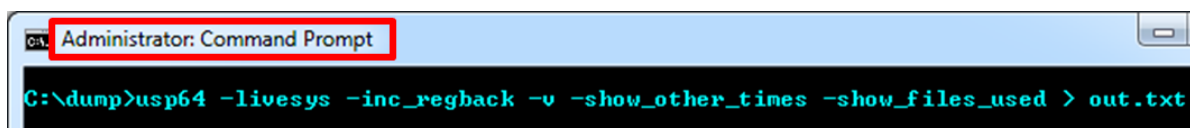
Basic Options
-csv                                         = output is comma separated value format
-csv12t                                     = log2timeline output
-v                                           = verbose, multiple lines per record
-sys "hive1 | hive2 | ..."               = use specified system hives
-user "hive1 | hive2 | ..."              = add user hive(s) to analysis
-setupapi "file1 | file2 | .."              = add setupapi file(s) to analysis
-sw "hive1 | hive2 | ..."                = add software hives to analysis
-amcache "file1 | file2 | .."              = *** add amcache file(s) to analysis
-evtx "file1 | file2 | .."                 = *** add eventlog(s) to analysis

Additional Options
-dateformat mm/dd/yyyy                     = "yyyy-mm-dd" is the default
-timeformat hh:mm:ss                       = "hh:mm:ss.xxx" is the default
-pair_datetime                             = *** combine date/time into 1 field for csv
-no_whitespace                             = remove whitespace between csv delimiter
-csv_separator "|"                         = change csv delimiter to a pipe char
-all_usb_devices                           = pull stats on all USB devices
-base10                                     = use base10 for evtx records numbers

New (Experimental) Options
-pipe                                       = *** pipe specially named files to process
-enumdir <dir> -num_subdirs <#>           = *** pull from folder files to process
-inc_regback                               = *** include primary and backup system hives
-show_other_times                          = *** show other times [uses -pair_datetime]
-show_files_used                           = *** show files contributing to USB device
-use_eventlogs                             = *** incl event logs for added times
-event_res <# secs>                       = *** group evtx/translogs by # sec intervals
```

3.1 Processing USB Artifacts from a Live Windows System

While the most difficult to implement, this use-case is the easiest to use. To run **usp** on a live Windows system, use the **-livesys** option to tell it to analyze the currently running registry hives and *setupAPI* logs.



```
Administrator: Command Prompt
C:\>usp64 -livesys -inc_regback -v -show_other_times -show_files_used > out.txt
```

Having administrator's access is required, since **usp** will need to take a snapshot of the appropriate hives on disk and start analyzing them. All output options are text and can be very large, depending on how many USB devices were plugged into the computer over the life of the system. Therefore, it is recommended to redirect the output to a file and analyze the output with a text editor.

The **-v** option is for verbose output, where each device found will contain multiple lines of data found for that device. The alternative format is one line per device, where the data is more useful to view in a spreadsheet. The **-show_other_times** tells **usp** to do any translation of timestamps embedded in various GUID data, as well as any additional timestamps that may have been found.

The **-show_files_used** identifies which artifact contributed data to the device. The **-inc_regback** tells **usp** to also include the backup registry hives in the analysis.

If desiring to add event log files to the **usp** results, one would need to add the option **-use_eventlogs**. This will cause **usp** to look for and parse the following logs: (a) *System.evtx*, (b) *Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx*, (c) *Microsoft-Windows-Kernel-PnP%4Configuration.evtx*, and (d) *Microsoft-Windows-Partition%4Diagnostic.evtx*.

```
"cmdline: usp64 -livesys -inc_regback -v -show_other_times -show_files_used"
USB Device: 0
Device name: Imation Pivot USB Device
Service: disk [usbstor]
vid/pid key update [UTC]: 08/17/2016 07:41:25.418
ven/prod/rev key update [UTC]: 08/17/2016 07:41:25.442
Disk Device update [UTC]: 08/11/2016 20:10:36.605
Volume Device update [UTC]: 08/11/2016 20:10:36.615
SetupAPI Log dates: [Local] [Install]: 08/11/2016 15:58:49.543; 08/11/2016 15:58:49.919;
DEVPKEY InstallDate [UTC]: 08/11/2016 19:58:50.508; 08/11/2016 19:58:49.919
Instance ID/Serial #: 3610a4a42ca185&0
Container ID: db16f407-665f-519d-bed8-eb2a02b37ef1
Volume ID: 13a6e427-5f16-11e6-adc9-00a0c6000021
Embedded time in Volume ID: 08/10/2016 16:18:30.500
Embedded MAC in Volume ID: 00-a0-c6-00-00-21
Disk ID: 13a6e425-5f16-11e6-adc9-00a0c6000021
Embedded time in Disk ID: 08/10/2016 16:18:30.500
Embedded MAC in Disk ID: 00-a0-c6-00-00-21
Volume name: PILOT
Volume name update [UTC]: 08/11/2016 19:58:52.130
Parent ID Prefix: <none found>
Vendor ID: 0718
Product ID: 0359
Revision: 1.20
Vendor/product: imation/pivot
USB hub/port used: 04/02
Acct that mounted vol: loan acct, on 08/11/2016 20:19:01.226 [UTC]
System hive details: C:\windows\system32\config\system [10/02/2016 02:53:46.224
Software hive details: C:\windows\system32\config\software [10/02/2016 03:24:48.0
ntuser hive details: c:\users\loan\ntuser.dat [10/02/2016 03:24:53.056 UTC] [md5:
SetupAPI log details: c:\windows\inf\setupapi.dev.log [md5:9b5a2467745327fc98da
USB Device: 1
Device name: WIBU - CodeMeter-Stick USB Device
Service: disk [vmusb]
vid/pid key update [UTC]: 08/17/2016 07:41:25.418
```

When opening the results file in notepad, a summary of the devices are listed with: device name, various timestamps, various identifiers, volume name, account name that mounted the device, and other miscellaneous data. The truncated diagram shows the output of the first USB device, which is labeled an “Imation Pivot USB Device”. The key timestamps are the original install date and the account that mounted the device, which should be the last time that user account plugged in the device. Other useful data includes the Instance ID/serial number, which *should* be unique for that device. I say *should*, since some vendors do not supply a unique number. However, from the empirical data, most vendors do try to honor the USB specification and embed this data into the device’s firmware.

The second way one can output the data is to display each USB device on its own row with the various metadata for that device as columns. To do so, one would use one of the field separated values, whether it is **-csv**, for Comma Separated Values, or some other delimiter. To use a different character for the delimiter, one can append the **-csv_separator <character to use>** option to the command, where one can force the delimiter to be a pipe character, comma character or tab. Since there is an issue that some USB device names may have a comma embedded into their name, **usp** tries to substitute any commas it sees in the names to spaces. Below is an example of the default **-csv** option:


```

usp - full ver: 0.24; Copyright (c) TZworks LLC
run time: 2013/09/30 17:01:03 [UTC]
cmdline: usp64 -livesys -dateformat yyyy/mm/dd -csv

device name,vid,pid, time-utc,install, time-local,disk dev, time-utc,vol dev, time-utc,type,vid,pid,hub,port
Kingston DataTraveler G3 USB Device,2012/03/30, 00:14:39.798,2012/02/28, 12:31:36.196,2012/03/13, 13:00:06.03
Kingston DataTraveler 2.0 USB Device,2012/03/30, 00:14:39.798,2011/11/15, 17:38:22.944,2011/11/15, 22:38:23
USB 2.0 USB Flash Drive USB Device,2012/04/04, 18:02:22.989,2012/04/04, 14:02:22.998,2012/04/04, 18:02:23.5
Kingston DataTraveler 2.0 USB Device,2012/04/20, 17:08:13.047,2012/04/16, 11:43:17.357,2012/04/20, 16:56:50
USB 2.0 USB Flash Drive USB Device,2012/04/29, 00:26:31.173,2012/04/28, 20:26:31.193,2012/04/29, 00:26:31.9
USB 2.0 USB Flash Drive USB Device,2012/06/11, 20:23:40.325,2011/09/11, 13:19:13.550,2012/06/11, 20:23:40.3
USB 2.0 USB Flash Drive USB Device,2012/07/13, 14:54:28.148,2012/04/16, 10:32:33.529,2012/07/13, 12:41:13.8
Kingston DataTraveler G3 USB Device,2012/08/05, 12:40:55.991,2012/02/03, 08:06:02.658,2012/08/05, 12:36:02.
Kindle Internal Storage USB Device,2012/09/14, 19:05:58.516,2012/09/14, 14:30:01.379,2012/09/14, 18:30:01.99
Kingston DataTraveler G3 USB Device,2012/10/20, 16:30:25.435,2012/10/20, 12:30:25.455,2012/10/20, 16:30:25.9
Kingston DataTraveler G3 USB Device,2012/10/20, 16:45:14.959,2012/10/20, 12:45:14.979,2012/10/20, 16:45:15.2
Kingston DataTraveler G3 USB Device,2012/10/26, 12:40:58.391,2012/10/26, 08:40:58.403,2012/10/26, 12:40:58.7
Kingston DataTraveler G3 USB Device,2012/11/12, 18:00:44.254,2012/07/09, 15:41:17.436,2012/11/12, 18:00:44.4
Kingston DataTraveler G3 USB Device,2012/11/29, 14:54:06.557,2012/03/12, 17:04:36.233,2012/11/29, 14:54:06.56
Kingston DataTraveler G3 USB Device,2012/12/06, 14:37:40.205,2011/10/11, 17:33:17.852,2012/12/06, 14:37:40.2
WDC WD80 0BB-00CAA1 USB Device,2013/01/14, 17:11:55.938,2013/01/14, 12:11:55.938,2013/01/14, 17:11:57.654,,
WDC WD32 00JB-00KFA0 USB Device,2013/01/14, 17:41:52.136,2013/01/14, 12:13:54.068,2013/01/14, 17:13:54.723
Kingston DataTraveler G3 USB Device,2013/01/29, 11:11:25,2013/01/29, 10:41:15,2013/01/29, 15:41:15

```

Using the CSV option with **-csv_separator "/"** syntax, one can output the data in CSV format using a pipe delimiter. The advantage of using the pipes as a delimiter is that pipes do not conflict with the USB names.

```

usp - full ver: 0.24; Copyright (c) TZworks LLC
License is authenticated and registered to David Tomczak; TZworks, LLC
run time: 10/08/2013 17:38:36 [UTC]
cmdline: usp -livesys -csv -csv_separator |

device name|vid|pid| time-utc|install| time-local|disk dev| time-utc|vol dev| time-utc|type|vid|pid|hub|port|vendor
GENERIC USB Mass Storage USB Device|| |09/21/2011| 20:45:28.142|09/22/2011| 00:45:28.922|09/22/2011| 00:45:29.328|
WD 5000AAV External USB Device|03/30/2012| 00:14:39.798|09/11/2011| 11:53:48.857|02/11/2012| 13:01:58.801|| |disk|
WDC WD16 001S-00MHB0 USB Device|03/30/2012| 00:14:39.798|02/18/2012| 11:15:50.229|02/18/2012| 16:15:50.956|| |disk|
Maxtor OneTouch USB Device|03/30/2012| 00:14:39.798|02/19/2012| 15:01:32.843|02/20/2012| 13:14:41.354|| |disk|#0d4
Kingston DataTraveler G3 USB Device|03/30/2012| 00:14:39.798|02/28/2012| 12:31:36.196|03/13/2012| 13:00:06.033|03/1
USB Flash Memory USB Device|03/30/2012| 00:14:39.798|09/14/2011| 18:20:14.749|02/14/2012| 20:31:11.713|09/14/2011|
Kingston DataTraveler 2.0 USB Device|03/30/2012| 00:14:39.798|11/15/2011| 17:38:22.944|11/15/2011| 22:38:23.886|11/
USB 2.0 USB Flash Drive USB Device|04/04/2012| 18:02:22.989|04/04/2012| 14:02:22.998|04/04/2012| 18:02:23.537|04/04
WDC WD16 00PEVS-00VAT0 USB Device|04/14/2012| 12:23:36.552|04/14/2012| 08:23:36.567|04/14/2012| 12:23:37.422|| |disk|

```

3.2 Handling the primary and backup hives

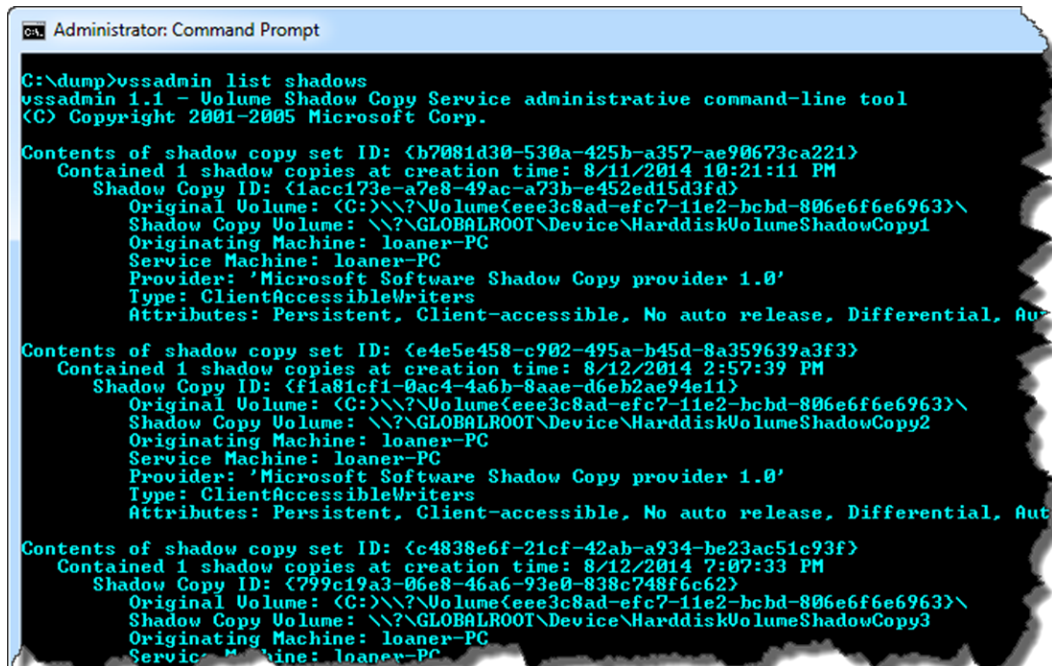
When running **usp** to look at a system volume, one can specify the **-inc_regback** option, to have **usp** process both the primary and backup hives in one report. This will result in **usp** trying to merge device artifacts with the same serial numbers and identifiable data.

3.3 Handling Volume Shadows from a Live System

Volume Shadow Copies of the system drive also contain artifacts necessary to perform USB analysis from a historical standpoint. By using the option **-vss <index of the volume shadow copy>**, **usp** can automatically pull the required hives and log data to generate a report on USB historical activity. Volume Shadow copies, as is discussed here, only applies to Windows Vista up to Win10. It does not apply to Windows XP.

To determine which indexes are available from the various Volume Shadows, one can use the Windows built-in utility **vssadmin**, as follows:

vssadmin list shadows



```
C:\>vssadmin list shadows
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

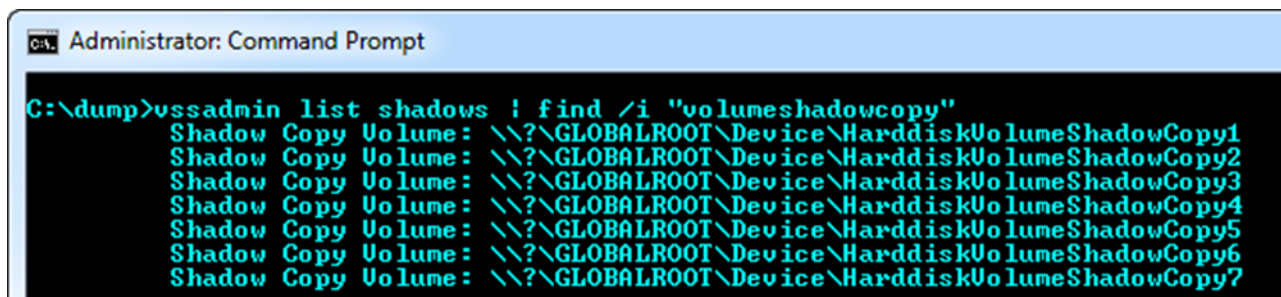
Contents of shadow copy set ID: {b7081d30-530a-425b-a357-ae90673ca221}
  Contained 1 shadow copies at creation time: 8/11/2014 10:21:11 PM
    Shadow Copy ID: {1acc173e-a7e8-49ac-a73b-e452ed15d3fd}
      Original Volume: {C:}\??\Volume{eee3c8ad-efc7-11e2-bcbd-806e6f6e6963}\
      Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
      Originating Machine: loaner-PC
      Service Machine: loaner-PC
      Provider: 'Microsoft Software Shadow Copy provider 1.0'
      Type: ClientAccessibleWriters
      Attributes: Persistent, Client-accessible, No auto release, Differential, Au

Contents of shadow copy set ID: {e4e5e458-c902-495a-b45d-8a359639a3f3}
  Contained 1 shadow copies at creation time: 8/12/2014 2:57:39 PM
    Shadow Copy ID: {f1a81cf1-0ac4-4a6b-8aae-d6eb2ae94e11}
      Original Volume: {C:}\??\Volume{eee3c8ad-efc7-11e2-bcbd-806e6f6e6963}\
      Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
      Originating Machine: loaner-PC
      Service Machine: loaner-PC
      Provider: 'Microsoft Software Shadow Copy provider 1.0'
      Type: ClientAccessibleWriters
      Attributes: Persistent, Client-accessible, No auto release, Differential, Au

Contents of shadow copy set ID: {c4838e6f-21cf-42ab-a934-be23ac51c93f}
  Contained 1 shadow copies at creation time: 8/12/2014 7:07:33 PM
    Shadow Copy ID: {779c19a3-06e8-46a6-93e0-838c748f6c62}
      Original Volume: {C:}\??\Volume{eee3c8ad-efc7-11e2-bcbd-806e6f6e6963}\
      Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy3
      Originating Machine: loaner-PC
      Service Machine: loaner-PC
```

To filter much of the unnecessary data to get to the index numbers, one can do the following:

vssadmin list shadows / find /i "volumeshadowcopy"



```
C:\>vssadmin list shadows / find /i "volumeshadowcopy"
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy3
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy5
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy6
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy7
```

This filters only the pertinent data needed to tell one which indexes are available for Volume Shadow copies. The number after the word *HarddiskvolumeShadowCopy* is the index that is used to pass as an argument into the *-vss* option.

3.4 Processing USB Artifacts from a 'dd' image of an NTFS disk

This use-case is broken up into 2 sections. The first addresses the situation if one acquires an image of an entire hard drive. The second addresses the situation if one images only a volume within a disk. Both

cases assume the acquisition of the image was a bit-for-bit copy, without using compression or some other proprietary format to store the final image.

For the first situation, one needs to find where the system volume starts. Specifically, one needs to identify to the **usp** tool what offset in bytes, from the start of the image, is the location of the system volume. The options that are used in **usp** are **-image <filename of image>** and **-offset <numeric value of start of volume>**. Both options need to be supplied for this to work.

To aid the user in doing this quickly, one can use **usp** in a two-step procedure. For the first step, **usp** will accept just the first option **-image <filename of image>** by itself, and then will analyze the image to see if there are any NTFS volumes on it. If it finds one or more, it will list their respective offsets. An example is shown below. The image is from a 40G drive that has one volume formatted as NTFS. By supplying just the image filename, **usp** displays to the user the offset of the NTFS volume it found, and then suggests what options to plug into the command line.

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The command entered is `C:\>usp -image W:\wd\public\image.drive.samples\images\wd.40g.img.001`. The output shows: `usp (USBSTOR parser) ver: 0.15; Copyright (c) TZWorks LLC`, followed by `volume offset [0x7e00] : ntfs`, and a suggestion: `use option [-image "W:\wd\public\image.drive.samples\images\wd.40g.img.001" -offset 0x7e00]`.

After the volume offset has been discovered, one can then proceed to the second step and supply this offset into the option **-offset <numerical value of start of volume>** to get **usp** to start scanning for the proper files it needs and outputting any USB statistics. Below is the final command based on the data provided by **usp** for the volume's offset. The output is redirected to a file, and the file is then opened in notepad.

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The command entered is `C:\>usp -image W:\wd\public\image.drive.samples\images\wd.40g.img.001 -offset 0x7e00 > usp.txt`. The second command entered is `C:\>notepad usp.txt`.

Looking at the output, one can see that there is a *Parent ID Prefix* present. This means that the operating system of the volume analyzed is pre-Vista, and in this case happens to be Windows XP. The first device listed is a 'SanDisk U3 Cruzer Micro USB' device, and from the data, was initially plugged in on 5/09/08. There were three user accounts that used this same device, as shown below. From the data, the last account to use the device was 'normaluser' on 07/01/09.


```
usp.txt - Notepad
File Edit Format View Help

USB Device: 0
Device name: SanDisk U3 Cruzer Micro USB Device
vid/pid key update [UTC]: 04/25/10 17:51:34.921
ven/prod/rev key update [UTC]: 04/25/10 17:51:35.015
Disk Device update [UTC]: 04/25/10 17:51:35.062
Volume Device update [UTC]: 04/25/10 17:51:35.578
Orig Install date [Localtime]: 05/09/08 11:24:13.000
Instance ID/Serial #: 00001889e57058ea&0
Driver: {4D36E967-E325-11CE-BF01-08002BE10318}\0009
Volume ID: eb4e26f0-1f26-11dd-bb6e-005056c00008
Disk ID: <none found>
Volume name: J:
Parent ID Prefix: 7&14671987&0
Vendor ID: 0781
Product ID: 5406
Revision: 4.05
Vendor/product: sandisk/u3_cruzer_micro
Acct that mounted vol: admin acct, on 05/23/08 04:13:16.875 [UTC]
Acct that mounted vol: dell acct, on 05/11/08 06:54:46.625 [UTC]
Acct that mounted vol: normaluser acct, on 07/01/09 02:59:50.500 [UTC]

USB Device: 1
Device name: ...
...
```

3.5 Processing USB Artifacts from a 'dd' image of an NTFS volume

This is the second type of situation that can be encountered with a 'dd' image. This is when the image is just of a volume versus the entire disk. For this case, the **-offset <#>** option does not need to be supplied and is assumed to be zero. Therefore, only the **-image <filename of image>** needs to be supplied. Below is **usp** processing an image taken from a Windows XP volume supplied by the SANS forensics 408 course. The example below is running **usp** in Ubuntu Linux. Again, one sees the *Parent Prefix ID* is present, confirming it is a pre-Vista image. The install timestamp, serial number of the device as well as the user account/time stamp that plugged the device into the computer is present.


```
testbox@testboxpc: ~/workarea/dev/usp/linux_vslick/package
testbox@testboxpc:~/workarea/dev$ ./usp64 -image xp_dblake.dd

USB Device: 0
Device name:          USB Device
vid/pid key update [UTC]: 06/30/07 22:41:57.208
ven/prod/rev key update [UTC]: 06/30/07 22:42:07.032
Disk Device update [UTC]: 06/30/07 22:41:57.528
Volume Device update [UTC]: 06/30/07 22:42:08.264
Orig Install date [Localtime]: 06/30/07 18:41:57.000
Instance ID/Serial #: 080909524d94e5&0
Driver:               {4D36E967-E325-11CE-BF01-08002BE10318}\0001
Volume ID:            49a30965-275a-11dc-a06c-800c0c446d64
Disk ID:               <none found>
Volume name:           <none found>
Parent ID Prefix:      7&cb28db9&0
Vendor ID:             1307
Product ID:            0163
Revision:              0.00
Vendor/product:        /
Acct that mounted vol: donald blake acct, on 06/30/07 22:42:10.026 [UTC]

USB Device: 1
Device name:          OLYMPUS u810/S810 USB Device
vid/pid key update [UTC]: 01/14/09 19:03:54.505
ven/prod/rev key update [UTC]: 01/14/09 19:03:58.030
Disk Device update [UTC]: <none found>
Volume Device update [UTC]: 01/14/09 19:04:00.023
Orig Install date [Localtime]: 01/14/09 14:02:54.000
```

3.6 Processing USB Artifacts off-line from extracted components

This is a situation where you have acquired a number of artifacts extracted from a Windows system, but don't have an image of the drive or volume. **usp** can handle this, if the user explicitly identifies which artifact is a system hive, which is a user hive, etc.

3.6.1 Specifying separate artifacts

If you only want to process a few artifacts and not an entire folder of files, one can invoke **usp** with the **-sys <system hive> -user "<user1 hive> | <user2 hive> | ... | <user# hive>" -setupapi <setupAPI log>**, etc, options. The syntax allows for one to include multiple hives of the same type by separating each similar hive with a pipe character. Below is an example of specifying discrete files and redirecting the report to the file named 'usp.txt'

```
Administrator: Command Prompt
C:\dump\usptest>usp -sys system -user ntuser.dat -setupapi setupapi.dev.log > usp.txt
C:\dump\usptest>notepad usp.txt
```

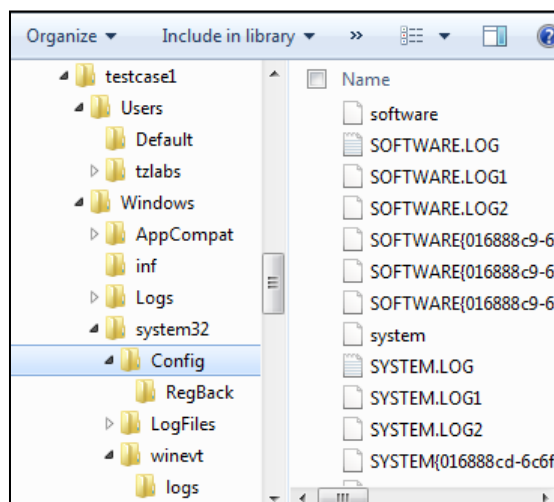
3.6.2 Specifying a folder of artifacts

Since **usp** can process many artifact files related to a USB devices used on a computer, sometimes it is just easier to put all the files require analyzing into a separate subdirectory and scripting the use of **usp** to process these files. Below is an example of one way to do this.

To collect the requisite files, one can use the **dup** utility from our website (https://tzworks.com/prototype_page.php?proto_id=37). This utility allows one to copy any file, or group of files from a live box or 'dd' image. This is especially handy for copying files when the operating system locks down the files, disallowing one to have even read access. Below we use dup to copy a group of files, including the registry hives, setupapi logs, and event logs.

```
Administrator: Command Prompt
C:\>dup64 -copygroup -pull_evtlogs -pull_reghives -out c:\dump\testcase1
```

The copied files are placed in the testcase1 subdirectory. The dup **-copygroup** option allows one to pull all the registry hives, including the user hives and system hives, as well as all the event logs. While it pulls more data than we need, it allows one to quickly grab the requisite artifact files needed for **usp** to process the USB devices that were plugged into this machine.



Now that the files are collected into a separate folder (testcase1), one can process all the artifact files using the **-pipe** command, like so:

```
Administrator: Command Prompt
C:\>dir c:\dump\testcase1 /b /s | usp -pipe -csv12t > out.csv
```

The above will look at all the files starting from the subdirectory *testcase1* and all subsequent subdirectories passing each file found into **usp**, for analysis. Even though there are files that are not pertinent for USB device data, **usp** will look at all files passed in and if it recognizes the file, it will then

try to parse it and merge any USB device data found into an overall report. In this case, we are formatting the report to be a Log2timeline CSV report named 'out.csv'.

If one cannot use the **-pipe** option, one can use the experimental **-enumdir** option, which has similar functionality with more control. The **-enumdir** option takes as its parameter the folder to start with. It also allows one to specify the number of subdirectories to evaluate using the **-num_subdirs <#>** sub-option.

3.7 Processing USB Artifacts from an externally mounted drive

This option is for the cases where you have an external hard drive and you want to analyze it without imaging it. In this case, one can put the hard drive under analysis in an external hard drive docking station with an interface to a write blocker and mount it as a separate volume. The syntax available allows one to access the drive in one of two ways, either as a mounted volume or as a mounted drive. The syntax for a mounted volume is: **-partition <volume letter>**. The syntax for a mounted drive is **-drivenum <drive #> -offset <system volume offset>**. The first is the easiest to use while the second forces the USB analysis to be directed to a particular volume offset. The output options are the same as in the previous use-cases.

3.8 Pulling USB Artifacts from a Monolithic VMWare NTFS image.

Occasionally, it is useful to analyze a VMWare image, both from a forensics standpoint as well as from a testing standpoint. When analyzing different operating systems, and different configurations, a virtual machine is extremely useful in testing out different boundary conditions. This option is still considered *experimental* since it has only been tested on a handful of configurations. Furthermore, this option is limited to *monolithic* type VMWare images versus *split* images. In VMWare, the term *split* image means the volume is separated into multiple files, while the term *monolithic* virtual disk is defined to be a virtual disk where everything is kept in one file. There may be more than one VMDK file in a *monolithic* architecture, where each monolithic VMDK file would represent a separate snapshot. More information about the *monolithic* virtual disk architecture can be obtained from the VMWare website (www.vmware.com).

When working with virtual machines, the capability to handle snapshot images is important. When processing a VMWare snapshot, one needs to include the parent snapshot/image as well as any descendants.

usp can handle multiple VMDK files to accommodate a snapshot and its descendants, by separating multiple filenames with a pipe delimiter and enclosing the expression in double quotes. In this case, each filename represents a segment in the inheritance chain of VMDK files (eg. **-vmdk "<VMWare NTFS virtual disk-1> | .. | <VMWare NTFS virtual disk-x>"**). To aid the user in figuring out exactly the chain of descendant images, **usp** can take any VMDK file (presumably the VMDK of the snapshot one wishes to analyze) and determine what the descendant chain is. Finally, **usp** will suggest a chain to use.

Below is an example of selecting the VMDK snapshot image file of *Win7Ultx64-000002.vmdk* (yellow box). Since the chain is incomplete, *usp* responds with what the dependencies are (shown in the red box), and then gives the user a suggested syntax to use for the command line to process this snapshot.

```

Administrator: Command Prompt

C:\>usp64 -vmdk E:\vmware\Win7ultx64\Win7Ultx64-000002.vmdk
usp full ver: 0.34; Copyright (c) TZWorks, LLC
License is authenticated and registered to David Tomczak; TZWorks, LLC
run time: 10/10/2013 16:26:31 [UTC]
cmdline: usp64 -vmdk E:\vmware\Win7ultx64\Win7Ultx64-000002.vmdk

=> vmdk file: E:\vmware\Win7ultx64\Win7Ultx64-000002.vmdk
    depends on parent vmdk "win7ultx64.vmdk"

suggested array: -vmdk "E:\vmware\Win7ultx64\Win7Ultx64-000002.vmdk ; E:\vmware\
Win7ultx64\win7ultx64.vmdk"
  
```

Repeating the command using the suggested chain of VMDK files, *usp* analyzes the chain, verifies it is valid, and if successful, outputs the results of the USB statistics for this snapshot of the NTFS volume.

4 Summary of all Options

The options labeled as 'Extra' require a separate license for them to be unlocked.

| Option | Description |
|----------------|---|
| <i>-csv</i> | Outputs the data fields delimited by commas. Since filenames can have commas, to ensure the fields are uniquely separated, any commas in the filenames get converted to spaces. |
| <i>-csvl2t</i> | Outputs the data fields in accordance with the log2timeline format. |
| <i>-v</i> | Verbose output. This option will output the parsed data as multiple lines for one record inputted. |
| <i>-sys</i> | Use the specified system hive during the parsing. Syntax is <i>-sys <system hive></i> . To specify multiple system hives, use the syntax: <i>-sys "<hive1> <hive2> ..."</i> . |
| <i>-sw</i> | Use the specified software hive during the parsing. Syntax is <i>-sw <software hive></i> . To specify multiple software hives, use the syntax: <i>-sw "<hive1> <hive2> ..."</i> . |

| | |
|---------------------|--|
| -user | Use the specified user hive(s) during the parsing. Syntax is -user <user hive> . To specify multiple user hives, use the syntax: -user "<user hive1> <user hive2> ..." |
| -setupapi | Use the specified setup API log during the parsing. Syntax is -setupapi <logfile> . To specify multiple user setup API logs, use the syntax: -setupapi "<log1> <log2> ..." |
| -amcache | Use the specified AmCache hive during the parsing. Syntax is -amcache <hive> . |
| -evtx | Use the specified event log(s) during the parsing. Syntax is -evtx <log> . To specify multiple evtx logs, use the syntax: -evtx "<log1> <log2> ..." . |
| -pipe | <p>This is an experimental option. Used to pipe files into the tool via STDIN (standard input). Tool will pull in all files first and begin parsing after last file is inputted. The set of files will be processed in one session. There are naming restrictions for the files that are processed. Specifically: (a) <i>System</i> hives must have the sequence of letters "system" in the name to be recognized as a system hive. (b) <i>Software</i> hives must have the sequence of letters "software" in the name to be recognized as a software hive. (c) <i>ntuser.dat</i> hives must have the sequence of letters "user" in the name to be recognized as a ntuser.dat hive; and (d) <i>setupapi.[dev].log</i> files must have the sequence of letters "setup" in the name to be recognized as a log file.</p> <p>As long as the each of the respective artifact files has the requisite sequence of letters, then any other letters can go before or after the sequence.</p> |
| -enumdir | Experimental. Used to process files within a folder and/or subfolders. Each file is parsed in sequence. The syntax is -enumdir <folder> -num_subdirs <#> . |
| -livesys | Pull USB stats on the current system |
| -inc_regback | Tells usp to also look at the backup system and software hives as well as the primary system and software hives. The output will merge the data appropriately. |
| -partition | Extract artifacts from a mounted Windows volume. The syntax is -partition <drive letter> . |
| -vmdk | Extract artifacts from a VMWare monolithic NTFS formatted volume. The |

| | |
|-----------------------|---|
| | <p>syntax is -vmdk <disk name>. For a collection of VMWare disks that include snapshots, one can use the following syntax:</p> <p>-vmdk "<disk1> <disk2> ..."</p> |
| -drivenum | <p>Extract artifacts from a mounted disk specified by a drive number and volume offset. The syntax is -drivenum <#> -offset <volume offset></p> |
| -image | <p>Extract USB artifacts from a volume specified by an image and volume offset. The syntax is -image <filename> -offset <volume offset></p> |
| -vss | <p>Experimental. Extract USB data from Volume Shadow to use for usp to parse into a report. The syntax is -vss <index number of shadow copy>. Only applies to Windows Vista, Win7, Win8 and beyond. Does not apply to Windows XP.</p> |
| -no_whitespace | <p>Used in conjunction with -csv option to remove any whitespace between the field value and the CSV separator.</p> |
| -csv_separator | <p>Used in conjunction with the -csv option to change the CSV separator from the default comma to something else. Syntax is -csv_separator "/" to change the CSV separator to the pipe character. To use the tab as a separator, one can use the -csv_separator "tab" OR -csv_separator "\t" options.</p> |
| -dateformat | <p>Output the date using the specified format. Default behavior is -dateformat "yyyy-mm-dd". Using this option allows one to adjust the format to mm/dd/yy, dd/mm/yy, etc. The restriction with this option is the forward slash (/) or dash (-) symbol needs to separate month, day and year and the month is in digit (1-12) form versus abbreviated name form.</p> |
| -timeformat | <p>Output the time using the specified format. Default behavior is -timeformat "hh:mm:ss.xxx" One can adjust the format to microseconds, via "hh:mm:ss.xxxxxx" or nanoseconds, via "hh:mm:ss.xxxxxxxxxx", or no fractional seconds, via "hh:mm:ss". The restrictions with this option is a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds. (Note: the fractional seconds applies only to those time formats that have the appropriate precision available. The Windows internal filetime has, for example, 100 nsec unit precision available. The DOS time format and the UNIX 'time_t' format, however, have no fractional seconds). Some of the times represented by this tool may use a time format without fractional seconds, and therefore, will not show a greater precision beyond seconds</p> |

| | |
|--------------------------|---|
| | when using this option. |
| -pair_datetime | Output the date/time as 1 field vice 2 for csv option |
| -all_usb_devices | The default behavior of usp is to try to pull data on those USB devices that store data. Typically, this are labeled as <i>USBSTOR</i> devices. If one wants to expand the data returned to be <i>all</i> USB devices, use this switch. Keep in mind every USB device that was connected to the computer will be displayed, including: HID (human interface devices, like mice and keyboards), USBHUBs, and other non-USBSTOR devices. |
| -base10 | Added for the evtx logs. The default behavior is to use hex for numbers. This says to use base10 for evtx records numbers and other related data in the Eventlogs. |
| -show_other_times | Experimental Option. This switch will display any additional timestamps found or derived. This option will also force the -pair_datetime option to allow rendering of multiple timestamps within a CSV field. Multiple timestamps in a field are delimited by semicolons. |
| -show_files_used | Experimental Option. This switch will display which artifact files contributed to a specific device report data. |
| -use_eventlogs | Experimental Option. Tells the tool to evaluate USB artifacts in the evtx type logs. This option is implied when using the -evtx or -pipe options. |
| -event_res | Experimental Option. When looking at evtx logs, group events into intervals separated by # seconds. The default is 2 seconds. This option also affects the SetupAPI.log data similarly. Syntax is: -event_res <# secs> |
| -utf8_bom | All output is in Unicode UTF-8 format. If desired, one can prefix an UTF-8 <i>byte order mark</i> to the output using this option. |

5 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The

license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

5.1 *Limited versus Demo versus Full* in the tool's Output Banner

The tools from *TZWorks* will output header information about the tool's version and whether it is running in *limited*, *demo* or *full* mode. This is directly related to what version of a license the tool authenticates with. The *limited* and *demo* keywords indicates some functionality of the tool is not available, and the *full* keyword indicates all the functionality is available. The lacking functionality in the *limited* or *demo* versions may mean one or all of the following: (a) certain options may not be available, (b) certain data may not be outputted in the parsed results, and (c) the license has a finite lifetime before expiring.

6 Conclusions

usp is an example of a command line tool used to automate gathering and reporting on USB device statistics for Windows operating systems. The tool can be run on either a live Windows system or in an off-line mode. The off-line mode has binaries that can run on Windows, Linux or Mac OS-X. As new Windows artifacts are discovered for USB statistics, they can be easily incorporated into **usp**'s existing, extensible architecture.

7 References

1. Windows Forensic Analysis DVD Toolkit, Harlan Carvey
2. Various forensic artifacts discussed in Computer Forensic Essentials from SANS Institute, <http://forensics.sans.org>
3. *TZWorks* LLC software libraries to parse various Windows' internals. www.tzworks.com
4. Various *Microsoft Technet* articles
5. VMWare Virtual Disk Format 1.1 Technical Note, www.vmware.com
6. SetupAPI logs examined include: *setupapi.dev.log*, *setupapi.dev.yyymmdd_hhmmss.log*, *setupapi.upgrade.log*, and *setupapi.setup.log*.
7. Eventlogs examined include: *Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx*, *Microsoft-Windows-Kernel-PnP%4Configuration.evtx*, and *Microsoft-Windows-Partition%4Diagnostic.evtx*.