A TagVault.org White Paper

TagVault.org
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08854
732.562.6031
www.tagvault.org

Analysis of Software Identification Tools
Review from May 2011

By Steve Klos
Executive Director, TagVault.org

December 2011

This page intentionally left blank

Contents

Executive Summary

Identifying what's installed on a computing device is similar to an archeological dig (i.e. trying to determine which software titles are installed based on various artifacts discovered on the device) and regularly results in incomplete and incorrect results. Unfortunately, compliance, logistics and security processes and procedures rely on this discovery data to manage an organization's infrastructure.

Incorrect data from software discovery utilities can and does result in:

- Higher risk of being out of compliance with software entitlements, which can lead to audit costs and further costs from the audit's outcomes.
- Higher risk of malicious, out-of-date or non-patched software being installed and used within the organization, with the associated potential for loss of data or intellectual property, or higher IT costs.
- Higher cost of specialized technical resources required to constantly create, monitor and update discovery procedures.
- Higher switching costs for discovery tools due to complexities and incompatibilities between tools.

There are many tools on the market that provide discovery capabilities with varying degrees of success depending on the platform, publisher and third party tools used. This paper reviews six mainstream discovery and identification tools and provides criteria to consider when selecting a tool.

It's important to know that the risks arising from incorrect software identification lie with the organization using the tool, not the tool provider. This means that any tool (or tools) an organization selects must use the most accurate information available – preferably information that comes directly from the software publisher. Unfortunately, there are too many publishers and tool providers for each to set up their own communication with every software publisher, and vice versa.

When it comes to larger organizations, software identification becomes even more difficult and expensive to manage. Most large organizations have multiple software discovery tools. These can be found in desktop management systems, help desk utilities, patch management tools, and software asset management tools. It is a good practice to be able to reconcile the data collected from these various tools to create a "single source of truth" about what is installed on the organization's computing devices, but the reality is that if the data from each of the tools is not provided in the same way, it is costly to consolidate this data into a normalized set of discovery data. This means that the reconciliation rarely happens, with the prime exception being during an external audit when it is beneficial for the auditor to provide additional justification for the final audit report. Even then, these reconciliation procedures often need to make use of pragmatic and industry-accepted estimates due to the cost and complexity of doing a complete reconciliation between data sets from different tools.

There is a better way that lowers costs for both the publisher and the buyer of software! ISO/IEC has published a software identification standard (19770-2) and TagVault.org is a non-profit program designed to make implementation of the standard and data consistency as easy and inexpensive as possible.

Read about the problems discovered through a live test of some these tools in this paper, then contact your software publishers and tool providers and require them to provide and support software identification (SWID) tags.

Some helpful reference documents are:

- Include SWID support in RFPs
- Require SWID through contract language
- Symantec's Statement of Support for SWID tags
- HP's DDMI Support for SWID tags

Evaluating Software Discovery and Recognition Tools

This white paper describes the results of an evaluation process which was applied to six software discovery and recognition tools. The intent of these tests was to identify how accurate and consistent these tools are when tested against a cleanly managed computing environment. Additionally, the test provided data used to determine how effectively data from one discovery tool could be validated and/or consolidated with data from a different discovery tool.

Testing for this paper did **not** include any attempt to create a "real-world" scenario where software has potentially been installed and removed multiple times. Additionally, there was no attempt to obfuscate or modify any discovery or installation data (a regular practice in larger corporations that customize their software installation packages).

Why Test Head-to-head?

There are two critical reasons to test software discovery tools in a head-to-head shootout. First, organizations must ensure that any tool they choose to use in their environment meets the specific needs of their organization. This may include the ability to accurately identify mass-market software from major software publishers, or may require more specific vertical identification capabilities. It may also include the need to add new titles into the recognition library used to match discovered data with software titles.

The second reason to test software head-to-head is to know what happens if multiple tools are used within your organization. Many larger organizations utilize different tools in different business units and need to consolidate data from these tools into a single configuration management database (CMDB) and/or have different tools collecting software discovery data for different IT processes. Having the ability to reconcile data across multiple IT processes ensures a higher degree of confidence that the collected data is correct and complete.

A side benefit of doing a head-to-head comparison and using the same client and server systems for the evaluation is that relative resource utilization for data collection and processing can be determined, although the purpose of this evaluation was simply to compare discovery results directly and no attempt was made to measure or track resource utilization or scalability of the various solutions.

Which Discovery Tools Were Included?

There were six tools used for this test. Unfortunately, due to the proprietary nature of these tools, their licensing requirements do not allow us to use company or product names in this review.

The types of tools used were:

- One tool is free to use, but not open source. This tool must be run on the device it is analyzing and is not designed to consolidate information across multiple devices. This is the only tool that has authorized the use of its name – it is WinAudit from Parmavex Services, and is identified as Tool 1.
- One tool is free to use and run on a single device, but it is part of a family of tools that include systems to consolidate information from multiple clients to a server database.
- One tool is free to use for up to 25 devices and can consolidate client data into a single database for reporting purposes.
- Two tools were run in a time-based evaluation mode. Both of these tools have additional capabilities for license compliance management.
- One tool utilized a remote inventory utility which was run on the various client demonstration systems with the inventory results processed on a production system. This tool is part of an overall suite of tools that includes compliance, desktop and security management utilities.

The various tools used in these tests can be found in practically any sized organization. In fact, due to the way engineering and IT teams tend to work, it's likely that your organization already uses (or has used) one or more of the tools tested in this evaluation.

Knowing how effective software identification tools are helps organizations design their testing and evaluation criteria.

What is Your Organization's "Source of Truth"?

Tracking software installations is extremely difficult to do properly. When it comes to software inventory/identification tools, most tools on the market today require that you **select only two** of the following three characteristics:

- Fast
- Somewhat accurate
- Inexpensive

The problem is that software is readily installed in many environments and may be automatically updated by the software itself (either through patches or software upgrades). New software is regularly introduced into organizations as old software is retired. Publishers regularly update existing products and release new products. They also frequently change names and titles between releases. IT organizations and third party consultants regularly update their own tools that they use to supplement the data in application recognition libraries.

Add to these difficulties, the fact that organizations often have multiple operating environments (Windows clients and servers, Macintoshes, UNIX, Linux and mainframe computers, and a huge number of mobile devices). Often, these environments require their own inventory utility which typically uses its own proprietary application recognition library.

These and other issues significantly decrease the accuracy of inventory and recognition processes and/or require that organizations spend significant resources in ensuring they know exactly what is installed. Remember that it's not just a case of identifying software known to be in use, it's also to identify security vulnerabilities such as software that may be added to an organization's infrastructure by end-users for personal or even malicious purposes.

To cope with the variables, many enterprises utilize a configuration management database (CMDB) to consolidate and track the hardware, software and other resources within their organization. If your organization has a system like this (or another environment that you use as the "source of truth" for software inventory – such as a discovery and license compliance or SAM tools) there are a number of questions you should be asking about the software inventory data used to populate these systems to determine how accurate your "source of truth" is:

1. How is software inventory data collected – is it done with a single tool, multiple tools, different tools for different geographies, etc.?

2. Is software identification data collected for all devices, or just for the devices running a specific operating system?

3. If inventory is collected with multiple different tools, how much effort is required to normalize data sets to ensure details such as publisher and software title are consistently reported?

4. What steps have been taken to validate that the software identification data is correct (for example, is there a second set of data that is compared to the CMDB to validate the data)? If your organization was audited, could it defend the accuracy of the organization's "source of truth"?

5. How often is the CMDB updated? Does your organization have real-time inventory that can be used for license optimization? Does your organization have the ability to quickly and automatically generate on-demand compliance reports that can be considered accurate and timely?

For organizations that have an effective process to manage their software inventory, the costs in terms of tools used, maintenance costs for the tools and engineering resources to ensure recognition data is up-to-date is typically quite high.

The focus of this particular evaluation process was not to specify the costs involved in consolidating or maintaining the data in the "source of truth" for your organization. Rather, this evaluation looks at the raw data that is likely being included in the organization's IT process for managing software. The cleaner and more accurate the input data, the faster, easier and less costly it will be to normalize this data in an effective CMDB and thus provide a defensible and auditable "source of truth" for the organization.

Why is Software Identification Important?

Software discovery and identification is required for a number of different IT processes. Obviously, having accurate identification information is critical to software license compliance activities. However, it is also critical to security management, desktop management, software policy enforcement and patch management activities (each of which may require their own software identification processes).

Additionally, today's software has become significantly more complex due to products being part of suites, bundles and various components that may come from different publishers all being installed as part of a single installation. Add to all of this the complexities of cloud and virtual operating environments and the explosion of mobile devices and accurate software identification is nearly impossible to do consistently today.

SWID tags based on the 19770-2 standard, especially if they follow the certification rules defined by TagVault.org, resolve these problems.

To be clear, even if publishers provide SWID tags today, there will still be a need for identification libraries of some type for the next few years until non-tagged software is no longer being supported. However, the commercial market is starting to recognize the value of SWID tags and products on the market today that are used for software identification are starting to generate SWID tags for legacy applications, reducing the reliance on identification libraries

Test Environment Setup

The testing environment was set up on virtual machines (VM). Four VMs acted as clients and one as the server. Each VM client device had exactly the same base operating system installed (Windows XP) with additional components installed as required for the particular test case.

Once each client device was set up and configured with the software required for each test case, the VM was cloned. This ensured that every tool evaluated had exactly the same configuration and installation data to work from.

Environment

Once each discovery and recognition tool captured inventory data from the clients, the data was exported to a standardized set of columns (typically using the SAM/IT tool of choice – Microsoft Excel). There was some basic data cleansing done on a few of the exported data sets – for example, if a tool did not have clearly separated fields for publisher and product title, these were broken apart and cleansed using some basic computational rules that could be reasonably applied.

Once the data was cleaned, all data was imported into a Microsoft Access database which included the device ID and Tool ID for every row of inventory data. Columns were added to the database as required if tools had additional data available for a particular inventory (for example, a few discovery and identification tools included some categorization information). The database was designed and expanded to support the capture of a superset of the data available from the various tools.

Analysis of the data sets was then done using Microsoft Access and Microsoft Excel. (Most images shown in this white paper are from Microsoft Excel.)

Client Devices

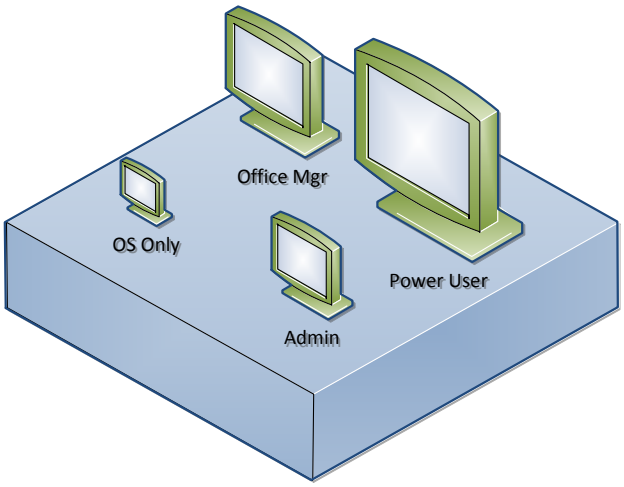
The devices were set up to have increasingly more complexity in the software installations.

Device 1 was considered a baseline system and did not have anything other than the Windows XP operating system installed with Service Pack 3 and all current patches.

Device 2 was targeted at a light administrative staff type user with some basic Microsoft Office and Adobe software installed.

Device 3 was targeted at an advance office manager with more Adobe and Office applications installed.

Device 4 was targeted at a power user or web developer with more Adobe software, a more complete Office installation, Visual Studio and other applications such as backup software, and additional web browsers.



Virtual Machine Test Bed

Software Applications Installed

The tests done were primarily focused on the discovery of software from two major vendors – Microsoft and Adobe – with a key focus on mainstream and current versions of the publisher's software. Additional titles were included as reference items that could be used to identify specific issues for any particular software discovery and recognition utility.

The software used for this test came from nine different publishers:

- Adobe
- Apple
- Dantz
- Drobo
- Google
- McAfee
- Microsoft
- Mozilla
- 7Zip/open source

From these publishers, there were effectively 22 separately licensable "products" used for the test. Note that from a security, logistics or software policy perspective, there were many more individual applications installed, all of which require tracking – this will be clear as you review the test results.

Annex A provides specifics on the applications used in the test as well as how the software was loaded on the four different devices so that you can get a feeling for the specifics of each device configuration.

Software Identification Analysis

There are numerous different perspectives that can be used to evaluate the software identification data provided during this evaluation. Your key criteria will typically be biased towards their job function. This review focuses in on some of the high level considerations, but you determine what aspects of software identification are most important to you and keep those priorities in mind as you complete your own product evaluations.

Organizations should recognize the benefits of having a consistent structure for software identification data. This identification data should be reconcilable between tools and provide a single source of truth for what is installed within the organization. The identification data needs to be in a structure and format that enables automation of compliance, logistics and security operations. If tools are used to manage the discovery process, an open, documented and accessible inventory database should be available through any tool used.

None of the tools tested provided a report that could be considered entirely accurate from a compliance, logistics or security perspective. Depending on your IT requirements, some tools may be "accurate enough", but it is clearly impossible for any tool that must determine software installations based on file and system settings to be 100% accurate in its software identification processing. This is due in part to:

- New software releases are introduced to the market daily
- Application recognition processes are based on "archeological techniques" – attempting identification based on various artifacts discovered on a computing device
- Application recognition is often based on inventory from third party tools that do not collect every piece of information required to accurately identify software
- Application and component relationships may not be something that can readily be identified by any inventory or recognition library

Let's review some of the specific issues seen during this tool evaluation.

Interpreting the Results

As mentioned earlier in this document, only one tool – WinAudit – has authorized the use of its name in this paper. All other tools are referred to as "Tool x" where x is a number. For every result provided in this paper, the relationship between the tool number and the tool itself remains consistent. WinAudit can be identified on the graphs by looking at the results for Tool 1. The data referenced as Tool 0 is based on the information that would be provided if SWID tags were utilized. Finally, this evaluation utilized six different tools, but seven data points are reported – the additional data point is based on the fact that one tool collects the information (with the data reported for the raw inventory) and then processes that data through an application recognition library. This results in two different data sets being available – both data sets are included separately.

Name Inflation

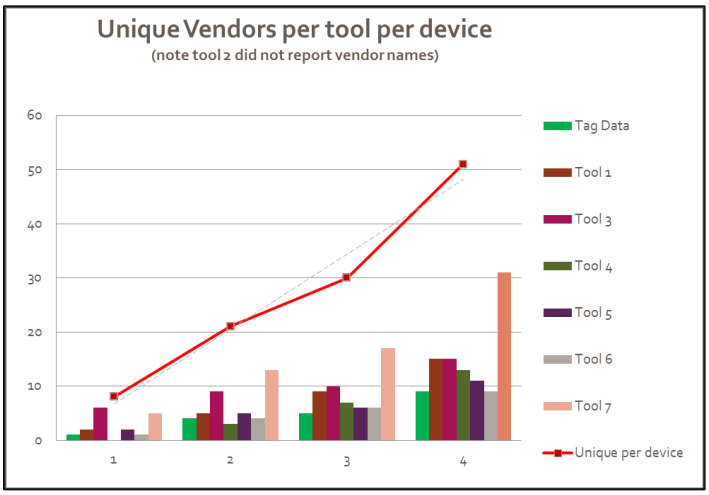
Often IT practitioners implement a software discovery process in their organization and quickly realize that the tools provide an excess of data. Filtering this data requires specialized knowledge and skills that often prove to be a daunting task for all but the most experienced technicians. Additionally with software releases occurring regularly, the filtering process requires constant monitoring and updates.

Larger organizations that use multiple tools, and those organizations that wish to have the ability to validate their data using a secondary data source in order to put their reports into a more defensible position, have an even more difficult problem. Since each tool uses a slightly different publisher name, product name or both, each of these names needs to be normalized and resources applied to ensure normalization stays up-to-date. The following section describes how the tested tools reported publisher and product names.

Publisher Names

In this evaluation the software installed on the devices came from nine different publishers. However, even for the base operating system with no software installed (Windows XP, SP 3 and MSIE 8) resulted in a report of between 0-6 different publishers listed (depending on the tool).

Across all tools, there were nine different unique publisher names discovered. As more software was added, the publisher name inflation increased dramatically as demonstrated in the graph below.



The red line denotes unique vendor names discovered across all tools for each specific device. For example, on Device 4 which should have had a total of nine publisher names listed (when focusing on licensed products), there were more than 50 unique vendors identified across the full set of tools tested. The results of a device in production use within an organization would likely show even more complexity.

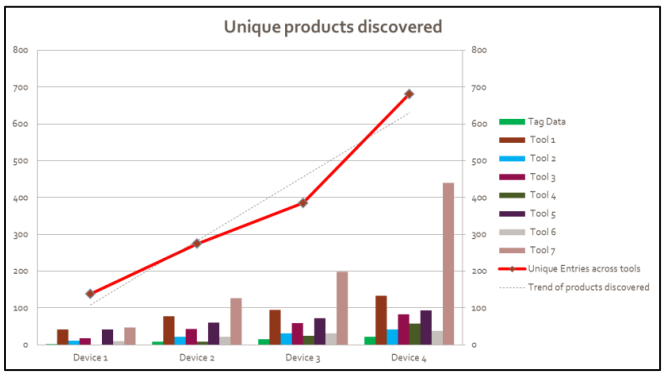
Looking at publisher names reported by each individual tool, it is clear that normalization is difficult to do properly. Adobe products were reported using eight different combinations (including "Adobe", "Adobe Systems", "Adobe Systems, Inc", "Adobe Systems, Inc.", etc.). If the only problem was that each tool reported these names differently, that would be one issue, but it turns out that only two tools attempted to normalize publisher's names at all, meaning that automated sorting and grouping in reports and analysis would be difficult and expensive to manage effectively.

Product Names

Product name inflation is a more difficult issue to deal with – primarily because new software products are being released regularly and every upgrade of a product introduces potential for the product name to change. There were 22 licensed software packages included in this test. This did not include the individual products that make up a suite product. (Note that a few individual Adobe Creative

Suite products were included in the test, as might be seen in a company where it is less expensive to purchase individual applications rather than a full suite.)

The results of this test are shown below:



In this case, it's easy to see how name inflation is a significant problem. Even in the basic set of software used for this evaluation, the test resulted in almost 700 unique product names across the various tools.

Dealing with name inflation is a significant cost to any organization. Filters must be put in place for a single tool and normalization rules must be created and maintained if reconciliation across multiple tools is desired or required.

If organizations instead rely on SWID tags, they receive registered vendor names and vendor-specified product names. This means that consolidating or reconciling names across different business units or different tools is a fully automated process (assuming the company and product name from the tag are used for reconciliation purposes).

Suite Identification Issues

There are many ways to collect and identify software installed on a specific computing device. With the variety of identification methods an organization must deal with, managing the identification and reporting of suites of application is extremely frustrating.

Software suites are groups of software that are sold under a single common license entitlement. Two common suites many organizations utilize are Microsoft Office and the Adobe Creative Suites.

Identification of the suite is often a problem by itself and is made worse by the fact that organizations often change or remove the product name that shows up in Windows Add/Remove Programs and the MSI database.

In fact, it is possible for an Adobe CS suite to be installed in such a way that the data in Add/Remove Programs and the MSI database indicates a completely different product is installed than what the computing device actually has installed and activated. Most software discovery tools will assume the Add/Remove Programs data is correct and therefore incorrectly report on the actual software installed on the device.

In the case of this evaluation, the Adobe Creative Suite Master Collection, Version 5 was installed in trial mode. The following table shows what each of the various tools reported.

Adobe Products – Device 4		
Product Name	Valid	Tool ID
Adobe Creative Suite 5 Master Collection	☐	1
Adobe Creative Suite 5 Master Collection	☐	2
Adobe Creative Suite 5 Master Collection (trial)	☑	3
Adobe Creative Suite 5 Master Collection	☐	4
Adobe Creative Suite 5 Design Standard 5.0	☒	5
Adobe Creative Suite 5 Master Collection	☐	6
Individual Adobe products listed	☒	7

Many organizations do not allow trial versions of software to be installed on production machines. So from a policy enforcement perspective, only one tool – Tool 3 – provided the information required to identify that this software does not meet the organization's policy requirements.

Since Tool 3 was the only tool that reported the fact that the software is installed in trial mode, it is also the only tool that would not over-report on installations of this particular Adobe title.

With a street cost for this software title of US $2,400, this type of mistake will cause a significant negative financial impact if the company believes it may be under-licensed, or worse, if the vendor were to audit the organization and base its audit on the inventory reported from any of the other software discovery tools tested in this evaluation other than Tool 3.

Note that tools 5 and 7 did not properly identify the software at all.

Tool 5 identified the individual applications that were installed and determined that the lowest cost version of the suite that included all the applications was the Design Standard Suite. This is a "best guess" approach to suite identification, but it is also a $1,300 mistake for each installation. Unfortunately, this is not atypical especially if the recognition library relies on a third party tool to capture inventory.

Tool 7 simply reported on the individual Adobe products that were installed and did not identify that a suite was installed.

Desired Identification Details for Suites

There are many reasons for an IT practitioner to know exactly what is installed on a device. Organizations need inventory data for compliance, license optimization, security, policy management, logistics, upgrade requirements, etc. To provide data that can be filtered appropriately for any of these purposes, the following summary of Adobe products is what the organization needs to collect.

In particular, the report above indicates that for this particular device, three separate licensable Adobe products are installed – Adobe Creative Suite 5 Master Collection, Adobe Flash Player and Adobe Shockwave Player.

For someone doing security reviews, they can instantly identify every application that was installed as part of the suite as well as any components that may be shared between multiple Adobe products (these are listed in the components section of the report).
This information can also be used by a practitioner doing license optimization to quickly identify that this particular installation is using a minority of the applications provided as part of the suite – in this case, depending on the organization's license structures, it would make sense to uninstall this version of the product from this device, and repurpose this installation for a power user while installing a copy of Adobe CS 5 Design Standard for this device (which costs US $1,300 less).

Adobe Creative Suite 5 Master Collection
+ **Applications**
| |-- **Photoshop CS5 Extended**
| |-- **Illustrator CS5**
| |-- InDesign CS5
| |-- Acrobat X Pro
| |-- Flash Catalyst CS5
| |-- Flash Professional CS 5
| |-- Flash Builder 4.5 Premium Edition
| |-- **Dreamweaver CS5**
| |-- Fireworks CS5
| |-- Contribute CS5
| |-- Adobe Premier Pro CS5
| |-- After Effects CS5
| |-- Adobe Audition CS5
| |-- Adobe OnLocation CS5
| |-- Encore CS5
| |-- **Bridge CS5**
| |-- **Device Central CS5**
| |-- Media Encoder CS5
+ **Components**
|-- **Adobe Air**
|-- **Extended Script Toolkit**
|-- **Extension Manager**
|-- **Adobe Community Help**
Adobe Flash Player
Adobe Shockwave Player

The information provided in the table can very readily be provided by the publisher using SWID tags at build time. Knowing which licensable products are installed, which applications form a particular suite and which shared components are installed on a device in a structured manner allows far more capable (and automated) processes for compliance, logistics and security.
It is simply not possible for third party utilities to accurately identify licensed products, the structure of the various components of a suite or bundle and the relationship of the various software components for every software title without relying on SWID tags.

Component Relationships

There are a number of reasons organizations need to know about "related" components that may be installed as part of a specific product installation.

Take a look at the components associated with the Adobe CS 5 Master Collection installation:

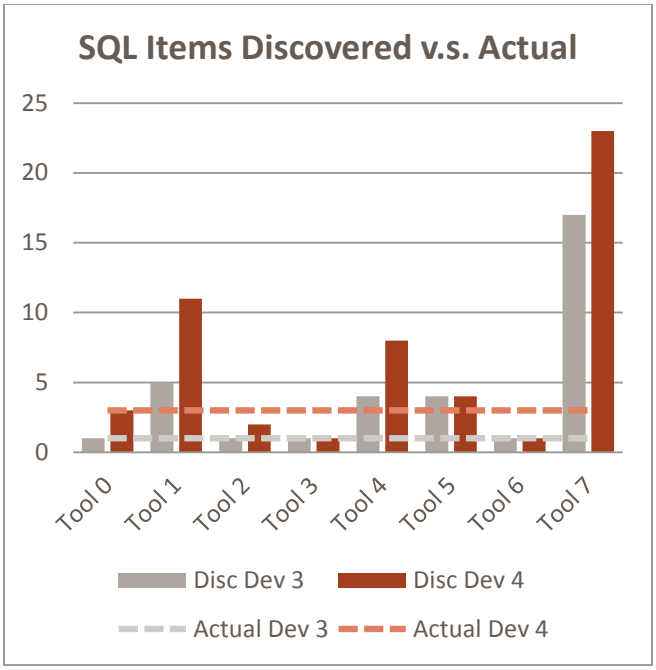
From a logistics/policy perspective, this information is critical. For example, if an organization decided that a component such as Adobe Air is not allowed to be installed on devices that are managed by their organization, they could very quickly review their tag data to identify exactly which applications would potentially be negatively impacted by that policy.

An even more relevant example is a policy that specifies that SQL Server cannot be installed on desktop or mobile devices unless explicitly required and approved. Organizations may want to apply this type of policy in order to ensure data stored in databases (which will often be related to corporate intellectual property) are properly backed up.

The problem in implementing this type of policy is how an organization identifies if there are SQL Server installations within the organization's computing device infrastructure. As well, knowing which product may have installed a version of SQL Server on a device helps determine if the installation is allowed.

Annex A shows that it's clear that a copy of SQL Server Express is installed if a device has Business Contact Manager installed. When doing this evaluation, it was expected that only one copy of SQL Server would be discovered on devices 3 and 4.

An initial view of the data would be used to determine how many SQL Server installations were on each device using each tool. This results in the following graph:



The "discovered" installations are shown as bar chart elements and the actual installations are shown with dotted lines.

It's clear just looking at the basic numbers that not one of the tools provided accurate details (remember, Tool 0 is the data set you would receive if SWID tags were used). It's important to realize at this point that the graph above is looking at a single computer (grey represents Device 3, red Device 4). Imagine what a real-world analysis across many tens of thousands of systems would identify.

However, the underlying data actually provides a wide variety of information due to the fact that there are many different versions of SQL Server. In this evaluation, the following table identifies what was installed on each device:

	Device 3	Device 4
SQL SVR Express Edition	1	2
SQL SVR Compact	0	1
SQL Server (any other edition)	0	0

Actual SQL Server Installation Details

The actual results reported by the discovery tools varied widely!

In fact, only two tools – WinAudit (Tool 1) and Tool 2 captured the correct editions and number of SQL Server installations on both devices. These tools did identify additional items that could likely, with review and constant updates, be filtered out in order to regularly come up with the correct SQL Server inventory. However, Tool 2 results are based on data captured as part of a raw inventory – the reports provided after processing the raw data through the tools application recognition library provided invalid results for both device 3 and device 4.

For Device 3, the tool results looked as follows:

	Total Discovered	# SQL Express Found (correct =1)	# SQL Compact Found (correct = 0)	Other SQL Editions
WinAudit	5	1	0	
Tool 2	1	1	0	
Tool 3	1	1	0	
Tool 4	4	0	0	SVR
Tool 5	4	0	0	SVR
Tool 6	1	0	0	SVR
Tool 7	17	---	---	---

Device 3 SQL Discovery Data Table

For this device, Tool 2 and 3 provided correct data when it comes to identifying SQL Express Edition. All other tools provided anomalies in the reported data. For Device 3, Tools 4, 5 and 6 reported that SQL Server was installed with no further qualifying information.

For Device 4, the tool results looked as follows:

	Total Disc	# SQL Express Found (correct =2)	# SQL Compact Found (correct = 1)	Other SQL Editions
WinAudit	11	2	1	
Tool 2	12	2	1	
Tool 3	1	2	0	SVR
Tool 4	8	0	1	
Tool 5	4	1	0	
Tool 6	1	1	0	
Tool 7	23	---	---	---

Device 4 SQL Discovery Data Table

Looking at the numbers for the SQL discovery data found on Device 4, one obvious issue is that Tools 5 and 6 mistakenly identify SQL Server Compact as SQL Server Express. This may not be a problem from a compliance perspective, but it could have more significant implications from a security or organizational policy perspective. Tools 4 and 5 had reported that SQL Server was installed on Device 3, but when two additional databases were included (an additional Express edition and a Compact edition), these tools stopped reported that SQL Server was installed.

Component Reporting Issues

The issues detailed in this section for the Adobe CS Suite and SQL Server edition are not surprising in the least. As the needs of the market change, software publishers often extend their software products to accommodate differing customer needs resulting in more options for editions, suites, etc. Additionally, the reuse of software modules by these publishers makes it extremely difficult for any third party tool (and often the tools provided by the software publisher directly) to properly identify software once it is installed on a device.

A few closing thoughts on this topic.

First, the complexities of software installations are making the compliance, logistics and security of computing devices significantly more difficult. Much of the focus of software identification has been focused on compliance and is primarily focused on Windows devices, but there are numerous IT processes where accurate and complete software identification data is required. It is critical that organizations have a correct and valid set of software identification information they can use for any and all IT

operations ranging from license compliance to security patch management to organizational policy compliance efforts.

Second, the only way to ensure accurate software identification data today is to require SWID tags to ensure organizations can identify relationships between software to allow IT practitioners to understand how a particular application or component ended up being installed on a device. A perfect example of this is the SQL Server versions installed on Device 4 – it is unclear why there were two versions of SQL Express edition installed on this device, without the contextual information of SWID tags, this type of relationship is often difficult if not impossible to understand.

Finally, many tools attempt to identify all software titles that require a license entitlement, but often fail to identify the individual applications that are allowed to be installed for a specific suite. This type of data is critical to manage and automate license optimization efforts, ensure organizational policy is applied and identify potential security threats. This data is critical to many IT operations, yet it is often not available, or is provided in a manner that swamps the practitioner with excessive data.

Results for Other Titles

Microsoft and Adobe products were generally the cleanest data from the inventory and discovery process because the tool providers have put a significant effort into cleaning data related to these companies. Lesser-known and less commonly used applications from smaller publishers tend to have "dirtier" data.

Invalid and Incomplete information

Software identification that is based on the collection of data from a computing device (especially, if the data collection is provided by a third party utility) will always be inaccurate and incomplete. This is due to the complexity of software identification and the fact that some software requires specialized information to be collected in order to differentiate multiple products.

In addition to the other details provided in this document, highlights of other issues identified during testing include:

- Although Microsoft Office 2007 Enterprise Edition was identified as being installed by all tools:
    - Only three of the tools identified which individual products were installed as part of the Office 2007 installation.
    - Office 2007 had 58 different unique names associated with it across all the tools.
    - One tool identified that a beta version of Excel 2007 was installed (it was not).
    - Business Contact Manager (BCM) was identified as being two different versions (either version 3.0, or as being for Outlook 2007) depending on the tool.
    - Two tools identified BCM as including a service pack, the rest did not.
    - One tool did not identify that BCM was installed at all.
    - None of the tools were able to identify that SQL Server Express was installed as part of the Business Contact Manager installation.
    - Although Microsoft Access was only installed on one of the test devices, one tool identified it as being installed on three devices. Only two tools identified Access as being installed on only one device.
- Visual Studio had a number of issues in the identification process as well, including:
    - Multiple tools included the registered copyright symbol ® in the name causing further issues with sorting, grouping and analysis.
    - Three tools identified that Business Objects Crystal Reports for Microsoft Visual Studio was installed, the other tools did not.
    - Visual Studio was identified using seven different names by the different tools.
    - One tool identified three different installations of Visual Studio using three different names.

- One tool identified a component of Business Objects installed, but did not relate that component to the Crystal Reports for Microsoft Visual Studio Installation.

- Even Microsoft's Internet Explorer had numerous identification issues, including:
  - Seven different versions of Internet Explorer were identified plus numerous entries that did not include a version.
  - Internet Explorer was identified using four different product names.
  - Some tools identified multiple different versions of Internet Explorer being installed (including version 2, version 8 and version 2009)
  - Some tools reported versions using commas to separate the versions, some used periods.
  - Some tools provided all four parts of the version information, others provided two, some provided no information on versions.
- Mozilla's Firefox was not identified as being installed by one of the tools. This is a problem if corporate policy is to run only a specified browser.
- Mozilla Firefox was reported as either "Mozilla Firefox", or "Firefox" with about an even split between these two names. This is a problem for reporting, grouping and analysis.
- Google Toolbar was completely missed by two discovery products and was reported using four different names by the other tools.
- McAfee Security Scan Plus was not identified by one of the tools and no tool provided any detail on what this product may be related to, nor how it was installed.
- 7Zip was identified using four different names with one tool identifying three different "products" related to 7Zip and two that are possibly related to 7Zip.

Some of these issues can be dealt with by having resources cleanse the data – this is particularly true of issues where too much data is provided. However, this takes time and resources with very specialized skills to create, test and update filters on a regular basis.

In many cases, if the data is unreported or incorrectly reported, it is not possible to correct.

The problems identified varied across the various tools tested. No tool was perfect in its discovery process and there is no expectation that any tool can be perfect if the process used is based on reverse engineering which product was installed based on the files and other system settings discovered on a computing device.

Until and unless publishers start to provide SWID tags with their products and tools use those tags for software identification, it will be difficult and expensive to get to a reasonably accurate estimate of what's installed within an organization's infrastructure. Until then, compliance, logistics and security processes will be working with incorrect and incomplete data!

Summary

Accurate software identification is nearly impossible to do through aftermarket solutions. A cottage industry has developed to provide "software recognition libraries" which to date has been the only reasonably automated solution to this problem. These libraries have essentially become checklist items for software asset management tools, so many of these companies have their own proprietary recognition library or have licensed a recognition library from a third party.

A key problem with the application recognition libraries is the scale of the problem they are trying to solve. The following are all potential sources of software that can be installed on a device:

- Tens of thousands of software publishers around the world
- Most of those publishers releasing multiple products and patches each year.
- Many of these publishers releasing software for multiple different operating systems

- Many publishers having multiple distribution channels (some of which have the contractual rights to change the name of the software)
- A tremendous number of hardware manufacturers and the drivers they create
- Internally or contract developed software
- A large and growing number of open source software developers

Adding to this is the fact that the number of platforms is exploding with the handheld, tablet, virtual systems and cloud-based software. Basically, it's impossible for any of the application recognition libraries to keep up with every software release and be able to properly identify them. Some tools are better with particular application types, publishers or platforms, but the sheer scale of the problem makes it impossible to get accurate results for all the software used in larger organizations.

The fundamental issue is that IT processes in the compliance, logistics and security areas require this data to be accurate in order to give accurate results. Software identification has to be owned by the publisher and provided in a standard manner. SWID tags provide that capability and they provide it today!

About TagVault.org

TagVault.org is a certification authority for software identification tags based on the ISO/IEC 19770-2:2009 standard. Formed as a non-profit organization under IEEE-ISTO, TagVault.org provides a shared library of software tools, technical knowledge and communications forums that decrease the costs of creating, managing and using software identification tags.

TagVault.org's certification process ensures tags fully conform to the specification while also ensuring that terms used in the tag are normalized. Certified software identification tags are digitally signed and time-stamped using a certificate issued by VeriSign —ensuring the accuracy of tag data that any third party can validate. Certified software identification tags lower the cost of software asset management for all SAM eco-system members.

For more information, please go to www.tagvault.org.

About IEEE-ISTO

IEEE-ISTO is the premier trusted partner of the global technology community for the development, adoption, and certification of industry standards. Its mission is to facilitate the life-cycle of industry standards development through a dedicated staff committed to offering vendor neutrality, quality support and member satisfaction. Fostering the market acceptance, adoption and implementation of standardized technologies, ISTO Programs span the spectrum of today's information and communications technologies. To find out more about ISTO, visit www.ieee-isto.org.

Phone: 732-562-6031
E-mail: info@tagvault.org
Web: www.tagvault.org

TagVault.org
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08854

Annex A: Client Configuration for Software Discovery and Identification Test

Device Configurations

As mentioned in the white paper, the devices were set up to have increasingly more complexity in the software installations.

Device 1 was considered a baseline system and did not have anything other than the Windows XP operating system installed with Service Pack 3 and all current patches.

Device 2 was targeted at a light administrative staff type user with some basic Microsoft Office and Adobe software installed.

Device 3 was targeted at an advance office manager with more Adobe and Office applications installed.

Device 4 was targeted at a power user or web developer with more Adobe software, a more complete Office installation, Visual Studio and other applications such as backup software and additional web browsers.

Software Applications Installed

Software for the test came from current and known software publishers. The goal was not to test the boundaries of the identification library, but rather to identity the basic capabilities that many IT environments would need to work within their environments.

The SOFTWARE used for this test came from nine different publishers:

- Adobe
- Apple
- Dantz
- Drobo
- Google
- McAfee
- Microsoft
- Mozilla
- 7Zip/open source

Depending on how the reader counts "software products", there were 22 known products (from a compliance and licensing perspective) that could be part of the installation. Breaking the CS 5 and Microsoft Office suites into their constituent parts would provide for significantly more software titles being identified.

Adobe Products	Microsoft Products	Other Products
Acrobat Reader	Business Contact Manager 2007	Drobo Dashboard
Acrobat Professional (trial)	Office 2007 Enterprise	Mozilla Firefox
CS 5 Master Collection (trial)	MSIE 8.0	Google Chrome
Flash	SQL Server Express Edition	Google Earth
Illustrator (trial)	Visual Studio	Google Toolbar
Photoshop Extended (stand alone, trial)	Windows XP	Apple iTunes
Shockwave		(Open Source) 7Zip
		Dantz Retrospect
		McAfee Security Scan Plus

The table below shows the software load configurations that were used for testing:

	Device 1	Device 2	Device 3	Device 4
Operating System	**Windows XP**	**Windows XP**	**Windows XP**	**Windows XP**
Adobe SW				Master Collection
				(only following installed)
		Acrobat Reader X	Acrobat Pro	------
				Dreamweaver
		Adobe Photoshop Ext	Adobe Photoshop Extended	Adobe Photoshop Extended
			Adobe Illustrator	Adobe Illustrator
		Flash	Flash	Flash
			Shockwave	Shockwave
Microsoft SW		Office 2007 Enterprise (specific apps installed)	Office 2007 Enterprise (specific apps installed)	Office 2007 Enterprise (specific apps installed)
		Word	Word	Word
		Excel	Excel	Excel
		Powerpoint	Powerpoint	Powerpoint
			Outlook	Outlook
			Office Tools	Office Tools
			Office Shared	Office Shared
				Access
			Business Contact Manager	Business Contact Manager
			SQL Svr Express	SQL Svr Express
				Visual Studio
Google		Google Toolbar	Google Toolbar	Google Toolbar
			Google Earth	Google Earth
				Chrome
Mozilla				Firefox
Apple				iTunes
7Zip Utilities				7Zip
Dantz				Retrospect
Drobo				Drobo Software

Note that additional software components were installed (or could be installed) as part of the installation sequences above – organizations that allow end-user installation of software may end up identifying more software than they expected without knowing how or specific titles were installed.

For example, **Microsoft SQL Server Express** and **Microsoft SQL Compact** editions were installed as part of the install process and the user was provided with the alternative to install McAfee products as part of the Flash installation and to install Google Chrome as part of the Google Earth installation process. Without an effective system that tracks every installation (or with the use of certified software identifications tags), the relationships between the various product installations would be difficult if not impossible to understand from an IT perspective.