Software ID Tags Support Better Cyber Security

Steve Klos, TagVault.org
John Richardson, Symantec Corporation

Abstract: Would you fly on an airline that did not verify its passengers’ identities, or that allowed high-risk passengers onto their flights? How about an airline that was unable to scan all checked luggage for known threats, or was unable to ensure that each piece of luggage was associated with a known, trusted passenger? This, unfortunately, is the position that IT organizations are placed into every day by their software applications. Software publishers do not typically provide secure, authoritative information that provides the critical information IT administrators require to validate the authenticity of their installed software applications and their executable files (program files, shared libraries, scripts, etc.). Typical computer systems such as a laptop with an operating system and a few software applications will have thousands of executable files installed on the system with no definitive way to authenticate these executable files, and to ensure that they have not been modified by any third party. To improve software supply chain security, IT organizations require standardized, authoritative software application information from software publishers that allows them to automate the following:

- Identifying all software applications, the operating system, their revision level, and all software updates and patches
- Associating all installed files to specific software applications, or to the operating system
- Validating that all installed software came from trusted software suppliers
- Validating the authenticity of each of the installed executable files

These capabilities are fundamental to securing computer systems. IT administrators must be able to automate these capabilities with a high level of confidence, and must be able to trust that their security tools can identify threats or known vulnerabilities quickly and definitively. However, without standardized, authoritative information provided by software publishers at the time software applications are released, it is remains very difficult for IT administrators to fully secure their computer systems.

The ISO/IEC 19770-2:2009 [1] Software Identification Tagging standard is a cross platform (Windows, UNIX, Linux, Mac) software identification data standard that provides the means for authoritative identification of software applications, operating systems, software updates, and patches. Tags also provide the means for:
- Validating the authenticity of install media
- Automating the authenticity validation of installed application executable files
- Automating the identification of installed applications with known vulnerabilities per the NIST National Vulnerability Database.

This article provides high-level information that outlines how software identification tags provide the fundamental building blocks required for building a resilient and automated IT cyber security ecosystem based on information that is very easily provided by software publishers.

Standardizing Software Identification in an IT Environment

Today’s software and computing environments are large and complex. Software applications are difficult to identify and track properly. Non-standard techniques employed by software publisher for software updates and patches make software identification even more difficult. In particular, it can be extremely difficult to determine the software revision level and whether or not a software update has been properly applied to a system. These complexities make it extremely difficult for IT organizations to ensure the most fundamental aspects of cyber security—mainly:

- Ensuring that all of the software deployed in the IT environment is authentic, unmodified, and from a trusted supplier
- Ensuring that all software is patched and that all known software vulnerabilities have been mitigated

This article covers the following four critical software supply chain security areas, and compares these areas to the security requirements of airline travel:

1) Authoritative Identification:
   Only authorized applications from trusted suppliers are installed (verified passengers).

2) Application Associations:
   Correct versions, patches and third party components are installed and related to their parent applications (all bags can be associated with passengers).

3) No Corruptions Allowed:
   Installed files or third party components have not been modified (no unknown passengers).

4) No Known Threats:
   Known application vulnerabilities have been resolved (passengers and bags do not have any known threats).

Software ID Tags—What Are They?

The missing link in today’s computing environment is standardized, authoritative information provided by software publishers that allow IT organizations to confidently automate a process that allows only trusted applications to be deployed in their environment; ensure that all application executable are authentic, and validate that the correct software updates have been applied. The International Organization for Standards (ISO) published the ISO/IEC 19770-2:2009 Software Identificat-
tion (SWID) Tagging standard enabling software publishers to provide standardized, authoritative, and secure application identification information in a consistent and secure format enabling security and asset management tools to authoritatively identify installed applications, components, and patches, and associate these items with files installed on a computing device.

This article does not provide detailed explanations of SWID tags—instead, it focuses on how authoritative software identification provided by SWID tags should be used to support other security standards and improve overall cyber security processes. For more detailed information on SWID tags, refer to the TagVault.org website [2].

SWID tags are not silver bullets that solve all cyber security concerns. Rather, SWID tags provide the necessary building blocks upon which the IT community can build a more secure infrastructure. Each of the following sections covers a different aspect of security capabilities supported by SWID tags. Providing SWID tags with the necessary data to provide all four critical functions ensures a much higher degree of authoritative and security related IT capabilities. Additionally, material covered in this article is focused on security of known software titles from trusted publishers. If a computing device includes unknown software installations, SWID tags can identify that fact, but will not provide more than the identification of unknown elements.

1) Authoritative Identification

When flying, a passenger must carry and present government issued identification documents. This provides a level of validation that the person carrying the document is who they say they are. This type of security relies on a trust model of a third party validating the identity of the passenger and some method to present that validation information to an unknown third party (a security screener validating a driver’s license for example).

When it comes to software, an organization that publishes software must validate who they are and that they have the necessary trust to “digitally sign” SWID tags. This is done through a certificate authority that validates that an organization is real and that it represents (i.e. owns) the organizational name.

Once a certificate authority certifies a digital certificate for a software publisher, the publisher can sign data with a private key that can be validated using the corresponding trusted public key as shown in Figure 1. SWID tags build on this trust model by allowing publishers to digitally sign a SWID tag that is associated with their software.

Signed SWID tags must also include a timestamp from a trusted timestamp server. This ensures two things—first, that the signed data cannot be modified by anyone (even the publisher) after the timestamp is applied without a third party being able to identify that data has been modified. Second, it allows a third party to identify that the
Supply Chain Risk Management

Figure 4: Diagram representing relationship of suite, applications, components and redistributable components.

signature was made during a time period when the certificate authority certifies the validity of the digital signature.

After a SWID tag has been digitally signed and timestamped, a third party can validate that the SWID tag was provided by an authoritative source during a time when the certificate authority indicated that the digital certificate was valid.

This level of trusted identification is required by any organization that needs authoritative data about the name of a software product, who published the product and which files the product installed. In short, this is the trusted identification information organizations must have to manage their IT environments securely.

Summary

Similar to passengers on an airline being required to identify themselves using government issued identification documents, software products require a similar level of identification. SWID tags that are digitally signed by the software publisher and include validation from a certification authority provide this trusted data.

Requiring a digitally signed SWID tags provides the ability for end-user organizations to validate that the software they have installed in their organization are the titles they expected and that they came from the publishers they expected and not some unknown publisher.

2) Application Associations

When an individual flies on an airline today, the airline must ensure that the owner of checked luggage travels on the same flight as the luggage. Providing the association of luggage to passengers is important to airline security as well as to the passenger who expects to have their luggage returned to them at the end of the flight.

A typical Microsoft Windows computer will have thousands of executable files (*.exe), and tens of thousands of shared libraries (*.dll, *.ocx, etc.) with a single application potentially responsible for the installation of hundreds of these executable files. Some of these files may be created and owned by the publisher, some may be redistributed versions of software from another publisher.

Today, it is nearly impossible to associate every executable file with its parent application. Additionally, if files from another publisher are redistributed, it is impossible for the publisher of the software to validate that the files in the redistribution package are authentic without support from SWID tags.

TagVault.org did an analysis of software identification tools using a very simple test case. The test utilized different installations of 22 currently supported, separately licensable products from nine different vendors. The test pulled results from six different software discovery tools. The graphical results in Figure 3 provide a clear indication why application associations are so critical to get right when dealing with software discovery data.

Basically the number of unique products discovered exploded from what would be a reasonable expectation of 22 different products to a total of 700 unique names across the various discovery tools. A large part of this problem has to do with the fact that there is no consistent method to identify software and the range of options produces wildly different collections of product names.

The keys from a security perspective require that a security operations manager needs to know:

- Which applications are related to a particular bundle or suite (for example, Word, Excel and PowerPoint are applications that could be stand-alone, or part of a Microsoft Office suite).
- Which components are related to a particular application (for example, Microsoft SQL Server Express is used by Microsoft Business Contact Manager which is a component of Outlook which is an application included in the Microsoft Office Suite).
- Validation that a redistributable component came from a known and trusted 3rd party and not some other, unknown, or unexpected entity (for example, was a C++ runtime library actually a redistributable library provided by Microsoft).

SWID tags provide these capabilities and more through the fact that each publisher provides their own digitally signed SWID tags. When a publisher such as Microsoft provides C++ runtime libraries in a redistributable package, they provide a digitally signed SWID tag indicating that the C++ runtime is owned by Microsoft and the vendor that is redistributing the runtime can reference this SWID tag as a “child” product providing the association. By using this method, security operations can readily identify parent/child relationships and be able to automatically group application and component data that would otherwise show up as independent applications.

Obviously, in highly secure environments, there will be a need to validate these relationships and the security of each item, but once validated the security policy rules can be applied to the inventory data collected from all devices in the organization.

Patches have a similar problem. When a patch is released by a publisher and the organization determines that it makes sense to install the patch, the system inventory can be used
To subscribe to CROSSTALK, visit www.crosstalkonline.org and click on the subscribe button.
to immediately identify every device that requires the patch. This is due to the fact that the patch’s SWID tag includes details for which software products it applies to. Obviously, validating that a patch is installed is as easy as checking that the SWID tag for that patch is installed. In highly secure environments, additional validations will be applied, such as ensuring that the publisher’s name for the patch matches the publishers name for the application being patched.

Summary

Just like airlines have the ability to associate a passenger’s checked bags with the passenger, software product and component relationships must be detailed by the publisher. These relationships need to be included even if software from a third party is included in an application’s installation routine. This is not difficult, nor is it expensive for publishers to provide, it simply has not been a requirement that software purchasers have made to their vendors.

Requiring the relationships between applications or components to be specified in SWID tags (including those that may be redistribute from a third party) ensures that security operations can validate the items are related to each other and that any redistributable components were, in fact, provided by the publisher indicated.

3) No Corruptions Allowed

Airlines must ensure that only passengers who have purchased tickets and who have gone through security screening are allowed on a flight. It is particularly important to validate that there are no unknown or high-risk passengers allowed on a flight.

This requirement extends to the files that are installed by a software product on a computing device. Security operations must ensure that the files installed on a device are not corrupted (i.e. unknown) or malicious (i.e. high risk) before installing and should be able to validate those details in real time.

There are two areas where file corruptions (regardless if they are an accidental or malicious) can occur and must be identified. The first is on the supply side—being able to validate that the software publisher’s distribution of a software installation package is exactly what the publisher shipped, and has not been modified by a man in the middle attack, is critical to secure systems. In these cases, a SWID tag with the complete installation file manifest that is digitally signed by the publisher and that includes secure hash values for every file is required. This allows an organization to validate that the files shipped by the publisher are exactly the same as the files received by the organization with no modifications.

The second issue dealing with file corruptions has to do with ensuring files that are installed on a device for a software product are known files that are not corrupted. In this case, the SWID tag must include a digitally signed file manifest that includes a secure file hash as part of the file list. Obviously, some files that are installed for a software title will be modified once installed (configuration options, data files, etc.)—and these files cannot include a trusted secure hash, however, at a minimum, the executable files for an application must be included in the manifest and must include a hash that can be validated.

Some software components (for example, Windows device drivers) need to be digitally signed for the operating system to trust them. Unfortunately, as was seen with the Stuxnet malware, simply requiring a digital signature for the operating system to trust a driver is not sufficient. In the case of Stuxnet, the drivers that were part of the malware payload were digitally signed, but were signed by a publisher other than the publisher that provided the software [6]. If the software had included a digitally signed manifest from the publisher, it would have been significantly more difficult for the Stuxnet malware to avoid detection as the changes it made to the application by replacing a core shared library with a malicious one, would have been detectable by ISO 19770-2 compliant scanning tools.

Summary

Just as airlines and security requirements would not allow unknown people or unscreened baggage onto a commercial flight, IT Security Managers should not allow unknown applications or executable files to be installed on their network. The details required to securely provide file manifests for software installation media as well as executable files that are installed on a device are not difficult or expensive to provide if the process to create the manifest is integrated into a product build cycle. The only organization that can do this in any cost effective manner is the publisher themselves.

Requiring SWID tags with secure file manifests to be included as part of the installation media (to ensure supply side security of the distributed files) as well as for the installed software allows security operations to validate applications, components and patches to any level of detail required.

4) No Known Threats

Airlines would not allow passengers to board with carry on or
checked luggage that contained dangerous materials. All baggage is screened to validate that potential risks to any flight are minimized.

The software environment in most IT operations is much more dynamic than is the case for a piece of luggage that can be screened once and assumed to remain safe as long as it remains in a secure environment. With higher complexity levels inherent in today’s software, publishers are regularly providing patches and updated configuration guidance to minimize the potential for security breaches. Unfortunately, the process by which patches and/or configuration changes can be identified as being required is often a very manual and time-intensive process.

Patches to software products must have the same type of secure file manifest provided so that as the patch changes an executable file, security monitoring systems can identify that a patch made the change and it is not due to a potentially malicious change to a file.

The requirement to include secure file manifests with applications, components, and patches provides a very positive side-effect that can be easily implemented for IT environments—organizations have enough data that IT processes can validate that any of these items are, in fact, installed on a device. If the SWID tag is installed on a device and it contains a secure manifest, a process can validate if the files are actually installed.

**Summary**

Just as all baggage destined to fly on a commercial airplane must be screened and declared safe, software must be validated to ensure it is up-to-date with no outstanding patches or configuration changes required. This can be done by ensuring that patches identify the software products they are targeted at as well as by providing secure file manifests to identify that a specified publisher provided the patch and that the files have not been modified.

Requiring patches to include SWID tags will not add to the complexity or cost of software development efforts if the procedures are integrated into the proper process of a patch build environment. The benefits to end-user organizations and tools that manage IT operations are very significant and can allow a much higher degree of security that simply is not possible today.

**Secure Software Requires Secure SWID Tags**

IT environments today are getting more and more complex. With this complexity, cyber security issues related to software installations, patches, and configurations are skyrocketing. With commercial organizations, national infrastructure systems and national security related systems becoming increasingly connected, it is clear that software must include a trusted and authoritative identification capability in a standardized, normalized format. The cross platform (Windows, UNIX, Linux, Mac) and cross vendor capabilities enabled by the ISO/IEC 19770-2:2009 Software Identification Tagging standard must be embraced by the software engineering community as a software assurance best practice for commercial and internally developed applications. Until the community makes these requirements, IT environments will continue to struggle with effectively managing and securing their software applications.

---

**ABOUT THE AUTHORS**

Steve Klos is the executive director of TagVault.org, a program of IEEE-ISTO. He is also the convener of ISO/IEC 19770-2:2009 and a member of the ISO/IEC JTC1/SC7 US Technical Advisory Group (TAG) and Work Group 21 (WG21 – targeting SAM Standards). Steve is also the recipient of multiple industry awards and certifications in Software Asset Management including being an IAITAM Fellow and a Microsoft Certified Professional with a Software Asset Management Competency.

Phone: 735-562-6031  
E-mail: stevek@tagvault.org

John Richardson has a background in software engineering, software engineering management, and software licensing. He has been involved with ISO Software ID Tagging since 2008 as the Symantec representative, helping to establish industry implementation standards and certification requirements, working with U.S. government agencies on approaches for integrating SWID tags and SCAP, and providing tools and process support to Symantec engineering teams as they integrate SWID tagging into their products.

Phone: 919-213-4027  
E-mail: john_richardson@symantec.com

**REFERENCES**

2. [http://www.tagvault.org](http://www.tagvault.org)