BUILDING BLOCK WHITE PAPER

SOFTWARE ASSET MANAGEMENT

Continuous Monitoring

V.2
This revision incorporates comments from the public.

David Waltermire
Information Technology Laboratory
david.waltermire@nist.gov

September 16, 2015

National Institute of Standards and Technology
U.S. Department of Commerce

NCCoE
NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE) at the National Institute of Standards and Technology (NIST) works with industry, academic and government experts to find practical solutions for businesses' most pressing cybersecurity needs. The NCCoE collaborates to build open, standards-based, modular, end-to-end reference designs that are broadly applicable and help businesses more easily align with relevant standards and best practices. To learn more about the NCCoE, visit http://nccoe.nist.gov. To learn more about NIST, visit http://www.nist.gov.

NCCoE building blocks address technology gaps that affect multiple industry sectors.

Abstract

Software asset management (SAM) is a key part of continuous monitoring. The approach described here is intended to support the automation of security functions such as risk-based decision making, collection of software inventory data, and inventory-based network access control. SAM, as envisioned in this project, uses a standardized approach providing a comprehensive, integrated view of software on the endpoint to support the following capabilities:

- publication of installed software inventory
- authorization and verification of software installation media
- software execution whitelisting
- software inventory-based network access control

At the core of this solution is the software identification (SWID) tag, an XML-based data format containing information describing a unit of software. A collection of SWID tags provides timely and accurate information about the current state of computing devices, also called endpoints. Organizations need to utilize this state information to measure the level of assurance of the software used to access organizational resources and to support critical business functions.

Automating SAM requires timely collection of software inventory data in the form of SWID tags and depends crucially on the trustworthiness of the SAM processes implemented for each endpoint. Secure transport protocols are required to enable SWID tag data to be exchanged. Trusted Network Connect (TNC) specifications provide the standards-based mechanisms to support the secure exchange of SWID tag information from and between computing devices.

Capabilities supporting this approach will be developed using existing commercial and open-source software with additional functional development as needed. As each capability is completed, it will be assessed against the original objective and this document will be revised to reflect relevant changes to the original approach.

KEYWORDS

access control; continuous monitoring; policy server; risk-based decision; security automation; software asset management; software identification; software inventory; visibility into endpoint

DISCLAIMER

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST or NCCoE, nor is it intended to imply that the entities, materials or equipment are necessarily the best available for the purpose.

COMMENTS on NCCoE DOCUMENTS

Organizations are encouraged to review all draft publications during public comment periods and provide feedback. All publications from NIST's National Cybersecurity Center of Excellence are available at http://nccoe.nist.gov.

Comments on this publication may be submitted to: conmon_nccoe@nist.gov.

ACKNOWLEDGEMENTS

The following people provided support to the development of the Software Asset Management building block:

Name	Organization
Valery Feldman	G2 Incorporated
Timothy McBride	NIST NCCoE
Greg Witte	G2 Incorporated

We gratefully acknowledge the contributions of all those who commented on the first edition of the Software Asset Management building block. Your comments helped make this a better document.

Table of Contents

1. EXECUTIVE SUMMARY

This document describes the technical challenge of collecting accurate and timely software inventory data, the desired security characteristics of a solution, and an approach using software identification (SWID) tags—a collection of data about software and its lifecycle and dependencies—and commercial, off-the-shelf technologies.

To build an effective security program, organizations need to know what software is running on their networks. Software asset management (SAM) can help organizations develop an inventory of installed software across their information technology (IT) networks, providing accurate and timely information about the current status of the software that accesses organizational resources and supports critical business functions. Software inventory in turn, supports the automation of security measures so that software running on business-critical systems can be routinely verified as authorized, not tampered with, and with vulnerabilities patched.

In many organizations, SAM processes are either manual or supported by a collection of disparate proprietary solutions. The approach to SAM described in this document addresses the technical challenge of collecting accurate and timely software inventory data using commercial, off-the-shelf products that are available to organizations of all sizes. We have employed a standardized approach that provides an integrated view of software and allows organizations to make risk-based decisions about their software vulnerabilities.

The core of this example solution is the software identification (SWID) tag, an XML-based data format describing a unit of software. A collection of SWID tags provides timely and accurate information about the current state of computing devices. Automating SAM also requires the secure exchange of SWID tag information between computing devices using the Trusted Network Connect (TNC) specifications, which provide the standards-based mechanisms.

This project was initiated in consultation with members of industry and other government agencies, who expressed a need for improved software asset management capabilities. An earlier draft of this document was made available for public comment, and those comments along with our responses are included at the end of the document. We invite readers to comment on this draft as well, so that the problem statement is as broadly applicable as possible before we begin work in NCCoE labs implementing model solutions. Please provide your comments to conmon-nccoe@nist.gov.

This project is part of a larger effort to show organizations how to implement continuous monitoring of their IT systems, and will result in a freely available NIST Cybersecurity Practice Guide.

2. Description

Goal

Continuous monitoring includes, but is not limited to, the monitoring of IT security and operational practices of asset management, configuration management, and vulnerability management. This building block—an NCCoE project that is applicable to multiple sectors—will demonstrate software asset management capabilities supporting continuous monitoring by focusing on accurate, timely collection of software inventory data and the secure exchange of software inventory data from and between computing devices. The software asset management functionality demonstrated by this building block may be used as part of a larger continuous monitoring capability supporting basic situational awareness of the software that is installed and in use on monitored devices.

In the context of this paper, the term 'situational awareness' represents timely collection and use of endpoint software installation state data that is collected using automated means. This includes software and patch inventory, software change data, and software footprint data (e.g., filenames, versions, hashes). This information is maintained by installers and other system processes used to manage the deployment of software (see Figure 1) and is communicated through standardized protocols (see Figure 2).

Background

Many, if not all, of an organization's mission or business essential functions—governance structure and core business processes—are dependent upon information technology. It is critical that organizations deploy solutions based on sound architectural approaches that support operational and security needs to protect the confidentiality, integrity and availability of information. Identifying and responding to new vulnerabilities, evolving threats and an organization's constantly changing security and operational environment is a dynamic process that must be effectively and proactively managed.

Continuous monitoring is defined as maintaining ongoing awareness to support organizational risk decisions[1]. Maintaining awareness of the software assets that reside on an enterprise network is critical to risk management and for defining the scope of authorization activities. A continuous monitoring system is composed of many different capabilities that support collection of security and operational data, analysis of real-time and historic data, and reporting of metrics in support of risk-based decision making at many different levels and contexts within an organization. To achieve this, a continuous monitoring system must provide visibility into organizational assets, awareness of

[1] NIST SP 800-137: Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations

threats and vulnerabilities, and support measurement of the effectiveness of deployed security controls.

A significant number of security controls relate to the management of software. These controls address the processes and technology required to successfully manage software throughout its deployed lifespan. Software is *released* by a publisher, *acquired* by an organization, *installed* by an administrator or user, *maintained* by applying patches (e.g., hot fixes, service packs) and updated software versions, and finally is *uninstalled* or retired when it is no longer of use or when the product reaches end-of-life. Throughout this lifecycle, a number of business processes are performed to manage the software. Licenses are tracked and purchased as needed as part of a license management process; software media is acquired as part of a supply chain; software is updated to take advantage of new features as part of a change management process; and patches are applied to fix security and functional flaws as part of vulnerability and patch management processes.

Automating SAM practices requires timely collection of software inventory data in support of ongoing awareness. SAM also supports disciplined network operations, change control, configuration management, and other IT and security practices. Tools supporting SAM help maintain an inventory of software installed and used on devices to access services and information maintained by an organization. Automating the management of software can be accomplished with a combination of system configuration, network management and license management tools, or with other special-purpose tools. SAM capabilities track the life cycle of an organization's software assets and provides automated management functions such as remote management of devices. The deployment and effective use of SAM capabilities is a key component of the implementation, assessment and continuous monitoring of software-related security controls such as those found in NIST Special Publication (SP) 800-53 Revision 4, ISO/IEC 27001:2013 Annex A , and other community-specific control catalogs.

Security Challenge

In order to support risk-based decision making and automated action, it is necessary to have accurate, timely information about the current state of computing devices, also called endpoints, to include the current state of software installed, authorized and used on each endpoint. Organizations need to utilize this state information to measure and sustain the level of assurance of the software used to access organizational resources and to support critical business functions.

The automated collection and secure exchange of software inventory data can further this assurance through automation systems that:

- provide an understanding of what patches and software updates are needed to ensure software vulnerabilities are minimized

- determine what software configurations need to be applied to ensure compliance with organizational configuration policies
- discover unauthorized installed software (or prevent the installation of unauthorized software)
- authorize the execution of software, preventing the execution of unauthorized or malicious code

In many organizations, SAM processes are either manual or are supported by a collection of proprietary solutions that do not scale for a variety of reasons. Often, proprietary solutions lack integration with other operational and security systems, are aligned with specific product families, and provide different informational views into the software they manage. As a result of implementing proprietary approaches, current SAM tools often don’t use information provided by the publisher to definitively identify and track software and its updates/patches.

This leads to significant issues, risks, and ongoing costs, such as:

- Current techniques are prone to errors in software identification and latency in support for new releases, and require on-going tweaking by the administrator.
- Software data is not normalized across tool sets making consistent, correlation and reporting difficult.
- Current tools cannot authenticate installation media using vendor-neutral methods resulting in implementation and deployment complexity, and often allow the installation of tampered software.
- Knowledge about the composition of installed software is not provided by most publishers as a common practice, making it difficult to detect unauthorized software modifications.
- Many software installation mechanisms do not associate installed software with dependent components (e.g., shared libraries, patches) in a way that is usable by software inventory and other software management tools, reducing the effectiveness of these tools.

SAM, as envisioned in this building block, requires a standardized approach that provides an integrated view of software throughout its lifecycle. Such an approach must support the following capabilities:

- Publication of installed software inventory – When connected to an authorized network, a device’s full or updated software inventory is securely reported to a central configuration management database that aggregates the software inventory of multiple devices for further analysis.
- Authorization and verification of software installation media - The ability to verify that the media is from a trusted publisher and that the integrity of the installation media has been maintained.

- Software execution whitelisting – The execution environment verifies that the software to be executed is authorized for execution and that the executable files and associated libraries have not been tampered with.
- Software inventory-based network access control – Control access to network resources at the time of connect based on published installed software inventory. Access to network resources can be limited if software is outdated or patches are not installed based on digital policies.

When used together, these capabilities enable enterprise-wide management of what software is allowed to be installed and executed. The collected information will also provide software version information to support license, vulnerability, and patch management needs. If historic software inventory information is maintained, retroactive analysis techniques can be applied on this data to determine historic vulnerable conditions in support of incident response and recovery processes. Finally, using collected software inventory, network access can be controlled, enabling the device to be connected to a remediation network, if necessary, so the appropriate software changes can be made before allowing it full access to the operational network.

The ability to support the intended business processes and the value obtained from automated collection and exchange of endpoint software inventory data depends crucially on the trustworthiness of the SAM processes implemented for each endpoint. At the very least, SAM processes must not undermine the trustworthiness of an endpoint by becoming a new avenue for attack. Therefore, SAM processes must leverage an appropriate set of security protections available on each particular platform to protect the confidentiality, integrity, and availability of software information. Since endpoints are highly variable in terms of available security protections, and since protection mechanisms should be increasing and improving all the time, it is neither practical nor desirable to establish a security threshold. Rather, the goal is for SAM processes to be flexible or configurable to take advantage of the best security features a platform has to offer.

3. Security Characteristics

The building block's SAM processes will:

- provide organizational visibility into endpoint software inventory supporting security and operational, risk-based decision making
- provide assurance that software installation media is authentic based on digital signatures and cryptographic hashes
- identify and support decision making related to software vulnerabilities prior to installation and during the lifecycle of installed software
- maintain a comprehensive, up-to-date view of the state of software installed on computing devices using one or more enterprise data stores

- uphold or improve the assurance of an endpoint's effective trusted computing base; endpoint SAM processes must not degrade an endpoint's security assurance

4. Approach

This building block focuses on the demonstration of SAM capabilities, based on standardized data formats and transport protocols. The general approach will address the following capabilities:

- verify the identity of the software publisher-provided installation media
- verify that installation media is authentic and hasn't been tampered with
- determine what software is installed and in use on a given endpoint device, including legacy and end-of-life products
- determine whether there is installed software on an endpoint that was not deployed using authorized mechanisms
- restrict execution of software that was not installed using authorized mechanisms
- identify the presence of software flaws in installed software
- enforce access control rules for network resources based on software inventory data

At the core of this solution is the software identification (SWID) tag, which is an XML-based data format containing a collection of information describing a unit of software. A SWID tag contains data elements that identify a specific unit of software and provides other data elements that enable categorization, identification and hashing of software components, references to related software and dependencies, and other data points. SWID tags can be associated with software installation media, installed software and software updates (e.g., service packs, patches, hotfixes). SWID tags associated with installation media (e.g., download package, DVD media) are called "media tags." SWID tags associated with software and associated software updates (e.g., patches) that have been installed are called "installation tags."

SWID media tags enable the associated media to be identified and verified using hash algorithms, and the publisher of the media to be authenticated using XML digital signatures containing an X.509 certificate.

Installation SWID tags managed by software installers or by system processes are responsible for describing, in a machine-readable form, the software and software updates that have been deployed to an endpoint. These tags are often organized in storage locations on the endpoint device. These tags enable installed software and software updates to be identified. Using this identification data, the relationship to software dependencies can be identified, the installation location to be found, and executables and other supporting files that are part of the installation can be identified and verified using associated version and hash information in the SWID tag's package footprint. Data pertaining to executable files can be used to verify executables at runtime, which partially supports whitelisting and blacklisting of application execution. Caution should be exercised when implementing runtime software footprint verification as part of a boot sequence for operating environments. Such capabilities may be necessary to ensure safe execution, but could also prevent execution of important system, maintenance or update processes.

Today, SWID tags are available for some commercially available software. Development of this building block should encourage additional commercial software vendors to provide additional SWID tagging support. For software that currently supports SWID tagging, support for SWID tagging will be expanded as needed. Additionally, SWID tags can be developed and deployed for custom software created by an organization, allowing this software to be managed using commodity software asset management tools. Third-party generation of SWID tags will be explored, which can be used to provide the data needed to manage custom or legacy products that do not have publisher-provided SWID tags.

Secure transport protocols are required to enable SWID tag data to be exchanged. The Trusted Network Connect (TNC) specifications provide the standards-based mechanisms to support the secure exchange of SWID tag information. The TNC standards enable accurate software inventory information to be made available to the enterprise. Using the TNC protocols, collected SWID tag data can be published to a data store managed by a policy server. This persisted information can be used to support configuration, vulnerability management, attack detection, network access control decision making, and other security automation tasks.

The building block's SAM capabilities, based on SWID tags and TNC transport protocols, will:

- allow installation media to be verified as authentic
- enable software execution to be limited to authorized software based on organizational policies
- demonstrate a standardized approach for securely collecting and exchanging software inventory data from networked endpoints, including those accessing a network remotely
- enable use of authoritative, vendor-provided SWID tag information to drive business processes
- make exchanged software inventory data available to operational and security systems where it can be evaluated against organizational policies supporting human-assisted and automated, risk-based decision making

The solution should conform to the Trusted Computing Group (TCG) Trusted Network Connect (TNC) Endpoint Compliance Profile (ECP) where possible. Data collection of

SWID tag-based software inventories must occur based on software installation change events. For the full value of this building block to be realized, both the SWID Tag and TNC ECP standards must be adopted by the SAM tools used.

Capabilities supporting the building block will be developed using existing commercial and open-source software with additional functional development as needed. As each capability is completed it will be assessed against the original objective and this document will be revised to reflect relevant changes to the original approach.

Gaps in technology and standards will be identified and solutions to these gaps will be proposed. Where practical, feedback will be provided to the standards development organizations to support revisions to the underlying standards.

The scope of the proposed solution is to demonstrate SAM capabilities, based on standardized data formats and transport protocols. The SAM building block focuses on the use of software identification methods for locally installed software applications and related installation/management processes. This document does not address the emerging examples of ephemeral software instances, such as cloud-based applications or other client-side active content technologies[2].

The use of ephemeral software brings additional security and asset management requirements; future iterations of this building block may explore management of active content as part of an overall software asset management solution. Additionally, this building block will investigate the appropriate means to use SWID tags for executable modules which might not be physically present on the local system, but may be accessible from network-based shares and removable drives; as well as, from software virtualization services.

The capabilities for this building block will be developed in the following manner:

Capability 0 – Establish SWID Tag Environment

The first capability prepares an environment for deployment and management of SWID tag data in the end-point device. It is a pre-condition for the other capabilities.

Development Approach

This capability will demonstrate three functions for supported platforms: a managed SWID tag installation environment, installer support for deploying SWID tags, and methods for tagging legacy software that have not been provided with a SWID tag by the software vendor.

[2] Client-side active content is described in NIST Special Publication 800-44: Guidelines on Securing Public Web Servers, version 2.

Management of Installed SWID Tags

This function will establish an environment on each endpoint platform for storage of installed SWID tag data as shown in Figure 1. During software installation, installers will deploy SWID tag information for the installed software to the SWID tag data store. This data store is typically the directory location identified by the SWID tag specification. For platforms that do not have an identified location, alternate storage mechanisms will be identified and used.

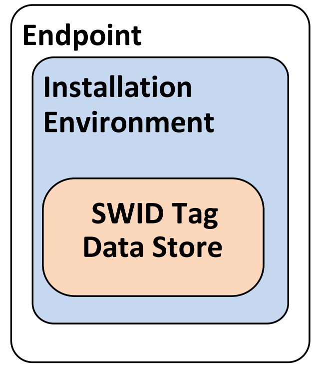

Figure 1 - Capability 0 Architecture

The development of this function will identify platform-specific security mechanisms to protect the SWID data from tampering and unauthorized access. Techniques will be employed to maintain and verify the integrity of stored data and limit access to read and modify SWID tags to authorized processes and users.

Installation environments will:

- limit write and modify access to the stored SWID tag data to software installation, inventory, and discovery processes
- limit read access to the stored SWID tag data to installation processes and other processes that are authorized to access SWID tag information

Deployment of SWID Tag Data During Software Installation

During software installation, the software installer is responsible for deployment of SWID tag information to the SWID tag data store. Development in this area will demonstrate that the appropriate capabilities are present in installers to manage the deployment and maintenance of SWID tags.

Installers will:

- deploy SWID tag data to the SWID tag data store for installed software and software deltas (e.g., patches, updates)
- clean up any legacy SWID tag data for software that is uninstalled or upgraded during the installation process.

Deployment of SWID Tags for Legacy Software

For software that does not have an associated SWID tag provided by the software vendor, it will be necessary to discover such software and to deploy or create an appropriate SWID tag. This function may be supported through the application of software patches that retroactively deploy a SWID tag for the patched software or by 3rd-party tools that provide this capability.

Outcomes:

- maintain an accurate accounting of installed software utilizing SWID tags
- uphold or improve the assurance of an endpoint’s effective trusted computing base; endpoint SAM processes must not degrade endpoint security assurance

Capability 1 – Publish Installed SWID Tag Data

The SWID tag information in an endpoint's SWID tag data store is useful to capabilities implemented on the endpoint. However, the ability to share this information with external capabilities enables the endpoint SWID tag information to support a variety of enterprise business, operational and security processes.

Development Approach

Prerequisite: Capability 0 – Establish SWID Tag Environment

Development of this capability will focus on using the transport protocols from the TNC standards to establish a secure channel between the endpoint and the policy server. Then SWID tag data for software installed on an endpoint can be used to securely communicate accurate software inventory to the policy server. This exchange between a SWID collector on the endpoint and a policy server receiving the published SWID tag data is depicted in Figure 2. Two modes of exchange must be supported: collector initiated publication of full or incremental SWID data and policy server initiated requests for specific SWID tag data.

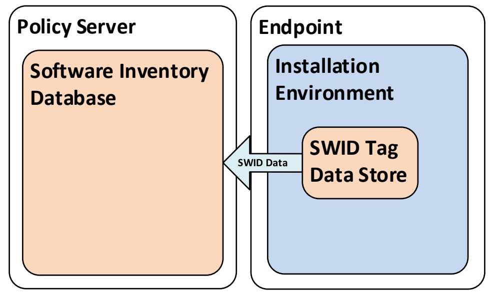

Figure 2 - Capability 1 Architecture

Regardless of the mode of exchange, the policy server will interact with the SWID collector on an endpoint device to access current and ongoing updates of SWID tag data. The policy server will maintain historic information for the software inventory of each endpoint it manages. Techniques will be identified to secure historic SWID tag data over the long-run.

The SWID collector will:

- support publication of SWID data based on the Endpoint Compliance Profile using the SWID Message and Attributes for IF-M specification which provides a standardized interface for messaging
- support publishing of full and incremental, event-driven SWID data to a policy server

The policy server will:

- receive exchanged SWID data
- store published SWID tag data for future retrieval, analysis, and possible automated or manual policy decision making and action

Outcomes:

- provide organizational visibility into endpoint device software inventory supporting security and operational, risk-based decision making
- enable identification of software with vulnerabilities throughout the lifecycle of installed software

- maintain a comprehensive, up-to-date view of the state of software installed on endpoints using an enterprise data store
- actively monitor software changes on one or more endpoints
- enforce enterprise policies based on missing patches or the presence of unapproved software
- provide support for other capabilities that are "downstream" processes (e.g., verification of configuration baselines related to specific software, vulnerability detection, patch management) that require enterprise knowledge of endpoint software inventory

Capability 2 – Media Verification Using SWID Tags

Media tampering is a significant attack vector presenting challenges for both software publishers and consumers. One of the benefits of a SWID tag is that it can be used to authenticate the publisher and verify the integrity of installation media. This enables install-time verification of the software media providing greater software assurance at the point of install.

Development Approach

Prerequisite: Capability 0 – Establish SWID Tag Environment

Development of this capability augments installation by enabling verification of installation media using a media tag. A media tag is a variant of a SWID tag that is bundled with the software installation media. The media can be an optical disk (e.g., DVD, BluRay), a shared network resource or a downloadable installation package. A media tag contains information that identifies the installation media, the software revision to be installed, and a file manifest containing paths and cryptographic hashes for each component of the software media. This collection of information can be signed using the XMLD Signature Syntax and Processing standard.

Processing of installation media by this capability requires incorporation of the SWID media tag in the installation media.

Installation environments will support:

- verification of the XML digital signature, including validating the certificate included in the signature based on a collection of available trusted root certificates
- verification of the installation media based on the file manifest and associated cryptographic hashes

Outcomes:

- provide assurance that software installation media is authentic based on digital signatures and cryptographic hashes
- verify the integrity of installation media prior to software installation
- enable the authorization of software installation based on the identification of the publisher and product

Capability 3 – Execution Authorization Using Installed SWID Data

The threat of many potential attack vectors is reduced by establishing greater trust that installed software has come through authorized channels. With this higher degree of assurance and verification that the software is trusted to perform as intended, policies such as whitelists can be used to limit software execution.

This building block capability will only be applicable to software with associated SWID tags that include footprint details. Absence of footprint details for software may be a policy item to consider as a part of this protection scheme. There is a desire to make this protection configurable so that policies may apply at the system, user, or process level.

This building block capability will also explore how SWID tags can help to enforce an authorized software list, such as a whitelist, that might be established by an organizational change management process.

Development Approach

Prerequisite: Capability 0 – Establish SWID Tag Environment

Development of this capability will utilize executable and shared library information defined in a SWID tag to allow or restrict program execution, based on an organizationally defined whitelist or blacklist. To support this, the execution environment will access installed SWID data, illustrated in Figure 3. These solutions will verify the integrity of the executable prior to execution using the cryptographic hash information associated with the executable in the installed SWID tag's package footprint. If this verification fails, then the execution will be prevented.

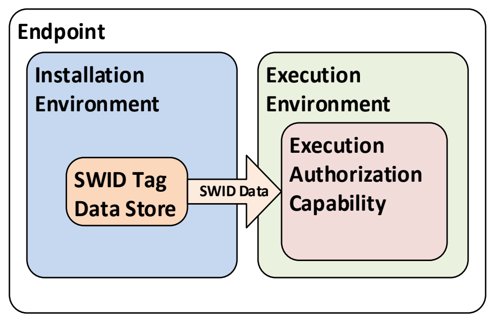

Figure 3 - Capability 3 Architecture

Additional policies may be employed to restrict execution privileges for specific users based on available SWID tag data. These policy expressions will use normalized software identifiers and metadata attributes in the SWID tag.

Outcomes:

- execution is restricted to software installed through authorized channels
- organizations define software execution policies based on SWID tag data
- policies are able to be defined and shared across multiple organizations, tools and processes

Capability 4 – Network-Based Policy Enforcement Based on SWID Information

Organizations ensure that the state of an endpoint is acceptable by controlling access to network resources at the time of connection and on an ongoing basis. Detecting and evaluating the software inventory of a device is an important dimension of network access control decisions.

Development Approach

Prerequisite: Capability 1 – Publish Installed SWID Tag Data

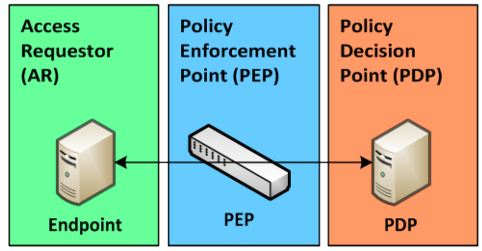

Figure 4 - Capability 4 Architecture

Development of this capability will use a policy server to make network access control decisions. Using published information collected from the endpoint, supported by capability 1, the policy server will authorize a computing device's connection to the network. The endpoint's software inventory will be monitored on an ongoing basis to detect software changes that violate network policy. If the endpoint's software inventory is found to be non-compliant at any point in time, the endpoint will be segregated for remedies to be addressed or disconnected.

Developed solutions will need to:

- establish TNC compliant infrastructure (e.g., policy decision point, policy enforcement point)
- implement network access control based on configured software usage and patching policy:
  - virtual local area network (VLAN) segregation of non-compliant hosts
  - patching on segmented VLAN

Solutions will support the following workflow:

1. When connecting to a network, the endpoint will discover the policy enforcement point.
2. The endpoint will publish full or updated software inventory using SWID data.
3. If the published software inventory is determined not to be compliant, access will be rejected or limited according to policy. If the endpoint is compliant, it will be granted access to network resources.
4. Endpoints will continue to publish changes to their software inventory on an ongoing basis while connected, allowing for compliance to be continuously measured.

Non-compliant endpoints will be handled according to the configured policy. If remedies can be applied, the following workflow will be supported:

1. The endpoint will be relocated to a remediation VLAN.
2. Patches will be downloaded and applied.
3. Non-compliant software will be requested for removal.
4. Once deficiencies are addressed, the endpoint will be re-verified and allowed access to the network.

Another supported variation will be to move the endpoint to a monitoring LAN with limited access if unapproved software is present.

Outcomes:

- prevent endpoints from accessing network resources if installed software is not compliant with software whitelist/blacklist or patch policy
- demonstrate support for a variety of mechanisms for remedy

Other Possible Capabilities

The demonstrable capabilities defined in this document represent areas where standards and product capabilities exist or are supportive of the solution. Additional capabilities may be added to the building block that address other requirements, building on these foundations. The SAM capabilities can be used with other security capabilities and tools that may be deployed at an endpoint or server to meet additional requirements. These may include dashboards that provide a network, enterprise, or organizational view of software inventory and software vulnerability information among other possibilities. Other avenues of collaboration will uncover new areas for expansion that will be added to the building block.

5. High-Level Architecture

The architecture for this building block, illustrated in Figure 5, depicts two distinct components: the policy server and the endpoint. The endpoint represents the computing device for which the software inventory is monitored. The policy server is the point of publication for software inventory data generated at the computing device. It is expected that multiple computing devices will interact with a single policy server. Organizations can also engage existing inventory management solutions to work with this building block to enhance the organizational view of software. For example, organizations may choose to implement multiple policy servers responsible for maintaining software inventory data for a network, office, data center or other organizational scope.

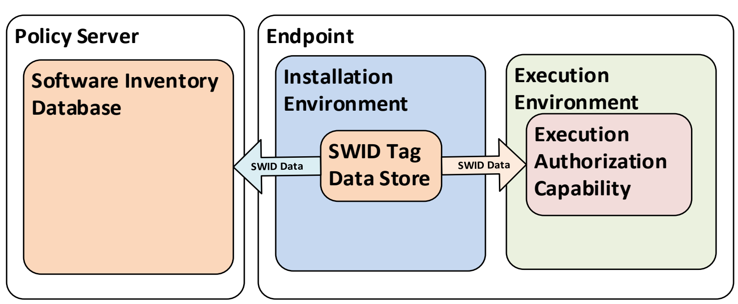

Figure 5 - Building Block Architecture

The diagram, illustrated in Figure 6, represents the TNC architecture that is used to transport software inventory data and to support network access control functionality supported by this building block.

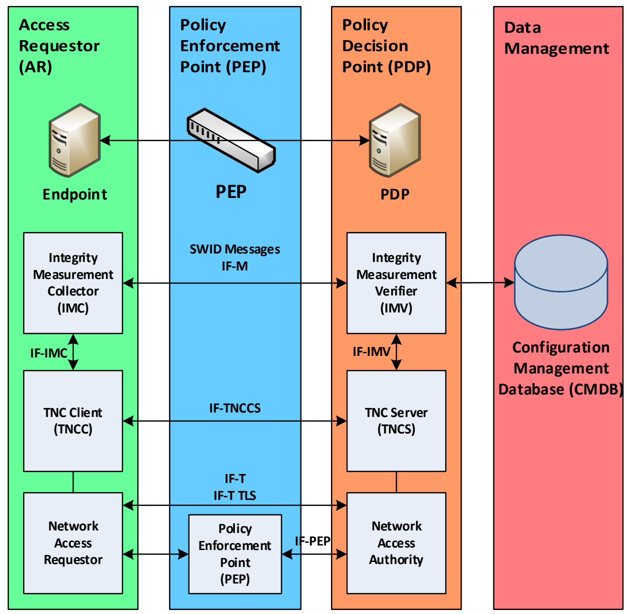

Figure 6 - TNC Architecture

In this diagram the endpoint is the access requestor (AR) and the policy server is the policy decision point (PDP). Access control is enforced by the policy enforcement point (PEP) which is typically a network device (e.g., wireless access point, switch, and firewall).

6. Relevant Standards

ISO/IEC 27001:2013, Information technology—Security techniques—Information Security Management—Requirements

ISO/IEC 19770-2:2009, Information technology—Software asset management—Part 2: Software identification tag

XML Signature Syntax and Processing (Second Edition), W3C Recommendation 10 June 2008

TCG TNC Endpoint Compliance Profile Version 1.0, Revision 9, 23 August 2013

TCG TNC SWID Message and Attributes for IF-M Version 1.0, Revision 14, 23 August 2013

TCG TNC IF-IMC Version 1.3, Revision 18, 27 February 2013

TCG TNC IF-IMV Version 1.4, Revision 8, 23 August 2013

TCG TNC IF-T: Binding to TLS Version 2.0, Revision 7, 27 February 2013

TCG TNC IF-TNCCS: TLV Binding Version 2.0, Revision 16, 22 January 2010

TCG TNC IF-PEP: Protocol Bindings for RADIUS, Specification Version 1.1, February 2007

TCG TNC PDP Discovery and Validation Version 1.0, Revision 9, 23 August 2013

7. Security Controls Mapping

The following table maps the security controls relevant to the SAM building block. It is intentionally over-inclusive including controls that contribute to and utilize the type of functionality enabled by SWID-aware software asset management. One should use the mapping to assist in evaluating implementations of the SAM building block and in deploying the building block within a broader IT security management regime.

Column one lists the security characteristic being described. Column two describes the example capability. The third column differentiates between controls that are enabled-by or contributed-to by SAM functionality. The purpose of this distinction is to indicate whether the SAM capability is essential to implementing this control or would assist in implementing the control. The fourth, fifth and sixth columns give the NIST Cybersecurity Framework Function, Category and Subcategory from the core controls list. The seventh and eighth columns give the crosswalk to IEC controls and NIST 800-53r4 controls from the Cybersecurity Framework Core crosswalk.

This exercise is meant to demonstrate the real-world applicability of standards and best practices, but does not imply that products with these characteristics will meet your industry's requirements for regulatory approval or accreditation

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Device security	Use SWID tags to support the inventory of devices and systems	Enables	Identity	Access management	ID.AM-1: Physical devices and systems within the organization are inventoried	ISO/IEC 27001:2013 A.8.1.1, A.8.1.2	NIST SP 800-53 Rev. 4 CM-8
Software inventory	Use SWID tags to support the inventory of software platforms and applications	Enables	Identity	Access management	ID.AM-2: Software platforms and applications within the organization are inventoried	ISO/IEC 27001:2013 A.8.1.1, A.8.1.2	NIST SP 800-53 Rev. 4 CM-8
System mapping	Map organizational data flows	Enables	Identity	Access management	ID.AM-3: Organizational communication and data flows are mapped	ISO/IEC 27001:2013 A.13.2.1	NIST SP 800-53 Rev. 4 AC-4, CA-3, CA-9, PL-8
System mapping	Use SWID tag capabilities to inventory external information systems	Enables	Identity	Access management	ID.AM-4: External information systems are catalogued	ISO/IEC 27001:2013 A.11.2.6	NIST SP 800-53 Rev. 4 AC-20, SA-9

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Software classification	Leverage tagging to prioritize resources	Enables	Identity	Access management	ID.AM-5: Resources (e.g., hardware, devices, data, and software) are prioritized based on their classification, criticality, and business value	ISO/IEC 27001:2013 A.8.2.1	NIST SP 800-53 Rev. 4 CP-2, RA-2, SA-14
Vulnerability identification	Utilize tagging to assist in the identifying of asset vulnerabilities	Enables	Identity	Risk assessment	ID.RA-1: Asset vulnerabilities are identified and documented	ISO/IEC 27001:2013 A.12.6.1, A.18.2.3	NIST SP 800-53 Rev. 4 CA-2, CA-7, CA-8, RA-3, RA-5, SA-5, SA-11, SI-2, SI-4, SI-5
Access	Use tagging to assist in the managing of physical access	Contributes	Protect	Access control	PR.AC-2: Physical access to assets is managed and protected	ISO/IEC 27001:2013 A.11.1.1, A.11.1.2, A.11.1.4, A.11.1.6, A.11.2.3	NIST SP 800-53 Rev. 4 PE-2, PE-3, PE-4, PE-5, PE-6, PE-9
Asset management	Use tagging to support the formal management of assets	Enables	Protect	Data security	PR.DS-3: Assets are formally managed throughout removal, transfers, and disposition	ISO/IEC 27001:2013 A.8.2.3, A.8.3.1, A.8.3.2, A.8.3.3, A.11.2.7	NIST SP 800-53 Rev. 4 CM-8, MP-6, PE-16

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Integrity verification	Leverage tagging to support integrity checking	Enables	Protect	Data security	PR.DS-6: Integrity checking mechanisms are used to verify software, firmware, and information integrity	ISO/IEC 27001:2013 A.12.2.1, A.12.5.1, A.14.1.2, A.14.1.3	NIST SP 800-53 Rev. 4 SI-7
Configuration management	Leverage tagging to support creation of an IT baseline configuration	Enables	Protect	Data security	PR.IP-1: A baseline configuration of information technology/industrial control systems is created and maintained	ISO/IEC 27001:2013 A.12.1.2, A.12.5.1, A.12.6.2, A.14.2.2, A.14.2.3, A.14.2.4	NIST SP 800-53 Rev. 4 CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10
Configuration management	Leverage tagging to support configuration change control	Contributes	Protect	Information protection	PR.IP-3: Configuration change control processes are in place	ISO/IEC 27001:2013 A.12.1.2, A.12.5.1, A.12.6.2, A.14.2.2, A.14.2.3, A.14.2.4	NIST SP 800-53 Rev. 4 CM-3, CM-4, SA-10

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Process improvement	Utilize tagging to support improvement of protection processes	Contributes	Protect	Information protection	PR.IP-7: Protection processes are continuously improved		NIST SP 800-53 Rev. 4 CA-2, CA-7, CP-2, IR-8, PL-2, PM-6
Process improvement	Utilize tagging to support protection effectiveness sharing	Contributes	Protect	Information protection	PR.IP-8: Effectiveness of protection technologies is shared with appropriate parties	ISO/IEC 27001:2013 A.16.1.6	NIST SP 800-53 Rev. 4 AC-21, CA-7, SI-4
Configuration management	Leverage tagging to support timely maintenance, repair and logging	Contributes	Protect	Maintenance	PR.MA-1: Maintenance and repair of organizational assets is performed and logged in a timely manner, with approved and controlled tools	ISO/IEC 27001:2013 A.11.1.2, A.11.2.4, A.11.2.5	NIST SP 800-53 Rev. 4 MA-2, MA-3, MA-5
Configuration management	Remote maintenance while preventing unauthorized access	Contributes	Protect	Maintenance	PR.MA-2: Remote maintenance of organizational assets is approved, logged, and performed in a manner that prevents unauthorized access	ISO/IEC 27001:2013 A.11.2.4, A.15.1.1, A.15.2.1	NIST SP 800-53 Rev. 4 MA-4

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Integrity verification	Utilize tagging to support the detection of malicious code	Enables, contributes	Detect	Continuous Monitoring	DE.CM-4: Malicious code is detected	ISO/IEC 27001:2013 A.12.2.1	NIST SP 800-53 Rev. 4 SI-3
Integrity verification	Leverage tagging to support the detection of unauthorized mobile code	Enables, contributes	Detect	Continuous Monitoring	DE.CM-5: Unauthorized mobile code is detected	ISO/IEC 27001:2013 A.12.5.1	NIST SP 800-53 Rev. 4 SC-18, SI-4. SC-44
Asset management	Leverage tagging to support the monitoring for unauthorized activity	Enables, contributes	Detect	Continuous Monitoring	DE.CM-7: Monitoring for unauthorized personnel, connections, devices, and software is performed		NIST SP 800-53 Rev. 4 AU-12, CA-7, CM-3, CM-8, PE-3, PE-6, PE-20, SI-4
Detection process	Use tagging to support definition of responsibilities	Contributes	Detect	Detection Process	DE.DP-1: Roles and responsibilities for detection are well defined to ensure accountability	ISO/IEC 27001:2013 A.6.1.1	NIST SP 800-53 Rev. 4 CA-2, CA-7, PM-14

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Detection Process	Detection Activities Comply with Requirements	Contributes	Detect	Detection Process	DE.DP-2: Detection activities comply with all applicable requirements	ISO/IEC 27001:2013 A.18.1.4	NIST SP 800-53 Rev. 4 CA-2, CA-7, PM-14, SI-4
Detection Process	Leverage tagging in testing detection processes	Contributes, Utilizes	Detect	Detection Process	DE.DP-3: Detection processes are tested	ISO/IEC 27001:2013 A.14.2.8	NIST SP 800-53 Rev. 4 CA-2, CA-7, PE-3, PM-14, SI-3, SI-4
Detection Process	Leverage tagging to support communication of detection information	Contributes	Detect	Detection Process	DE.DP-4: Event detection information is communicated to appropriate parties	ISO/IEC 27001:2013 A.16.1.2	NIST SP 800-53 Rev. 4 AU-6, CA-2, CA-7, RA-5, SI-4
Detection Process	Leverage tagging to improve detection processes	Contributes, Utilizes	Detect	Detection Process	DE.DP-5: Detection processes are continuously improved	ISO/IEC 27001:2013 A.16.1.6	NIST SP 800-53 Rev. 4, CA-2, CA-7, PL-2, RA-5, SI-4, PM-14

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Analysis Process	Utilizing tagging in investigating notifications from detection systems	Contributes	Response	Analysis	RS.AN-1: Notifications from detection systems are investigated	ISO/IEC 27001:2013 A.12.4.1, A.12.4.3, A.16.1.5	NIST SP 800-53 Rev. 4 AU-6, CA-7, IR-4, IR-5, PE-6, SI-4
Analysis Process	Utilize tagging to support the analysis and understand of incident impact	Contributes	Response	Analysis	RS.AN-2: The impact of the incident is understood	ISO/IEC 27001:2013 A.16.1.6	NIST SP 800-53 Rev. 4 CP-2, IR-4
Analysis Process	Use tagging to support the utilization of forensics	Enables, Contributes	Response	Analysis	RS.AN-3: Forensics are performed	ISO/IEC 27001:2013 A.16.1.7	NIST SP 800-53 Rev. 4 AU-7, IR-4
Analysis Process	Categorize Incidents	Contributes	Response	Analysis	RS.AN-4: Incidents are categorized consistent with response plans	ISO/IEC 27001:2013 A.16.1.4	NIST SP 800-53 Rev. 4 CP-2, IR-4, IR-5, IR-8
Mitigation Process	Use tagging to support the containing of incidents	Enables, Contributes	Response	Mitigation	RS.MI-1: Incidents are contained	ISO/IEC 27001:2013 A.16.1.5	NIST SP 800-53 Rev. 4 IR-4

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Mitigation Process	Use tagging to support the mitigating of incidents	Contributes	Response	Mitigation	RS.MI-2: Incidents are mitigated	ISO/IEC 27001:2013 A.12.2.1, A.16.1.5	NIST SP 800-53 Rev. 4 IR-4
Mitigation Process	Use tagging to support the mitigation or accepting of new vulnerabilities	Enables, Contributes	Response	Mitigation	RS.MI-3: Newly identified vulnerabilities are mitigated or documented as accepted risks	ISO/IEC 27001:2013 A.12.6.1	NIST SP 800-53 Rev. 4 CA-7, RA-3, RA-5
Process Improvement	Update response plans	Contributes	Response	Improvements	RS.IM-1: Response plans incorporate lessons learned	ISO/IEC 27001:2013 A.16.1.6	NIST SP 800-53 Rev. 4 CP-2, IR-4, IR-8
Process Improvement	Update response strategies	Contributes	Response	Improvements	RS.IM-2: Response strategies are updated		NIST SP 800-53 Rev. 4 CP-2, IR-4, IR-8
Recovery Process	Execute recovery plan	Contributes	Recovery	Response Planning	RC.RP-1: Recovery plan is executed during or after an event	ISO/IEC 27001:2013 A.16.1.5	NIST SP 800-53 Rev. 4 CP-10, IR-4, IR-8
Process Improvement	Adapt recovery plans	Contributes	Recovery	Improvements	RC.IM-1: Recovery plans incorporate lessons learned		NIST SP 800-53 Rev. 4 CP-2, IR-4, IR-8

Security Characteristic	Example Capability	Enables, Contributes	CSF Function	CSF Category	CSF Subcategory	ISO/IEC	NIST 800-53 rev4
Process Improvement	Update recovery strategies	Contributes	Recovery	Improvements	RC.IM-2: Recovery strategies are updated		NIST SP 800-53 Rev. 4 CP-2, IR-4, IR-8

562

8. Component List

- network infrastructure devices (e.g., routers, switches, firewalls)
  - vendor provided
  - either physical or virtualized
- operating system virtualization cluster
  - various operating system installations (e.g., Windows, OS X, Linux)
  - virtualization hardware
  - virtualization stack
- application software
  - Policy server
  - Policy enforcement point
  - Policy decision point
  - Software with SWID tags

9. Comments

We received 21 comments regarding the draft building block. The following listing in this section includes a brief summary of each comment and the associated response. Where necessary, we have revised the building block accordingly.

1. This document should clearly identify that many current SAM tools use proprietary techniques and are not using information provided by the publisher to definitively identify and track software and its updates/patches. This leads to significant issues, risks, and ongoing costs such as:

   - Current techniques are prone to errors, latency in support for new releases, and require on-going tweaking by an administrator;
   - Data is not normalized across tool sets making consistent, centralized reporting difficult;
   - Current tools cannot authenticate installation media and installed files using standard data for each software release and for patches and updates;
   - Often necessary software metadata is not provided by publishers as a best practice;
   - Many tools are unable to associate installed software with dependent components, patches, etc.; and
   - Current approaches don’t scale.

   Response: Text was added to the third and fourth paragraphs in the Security Challenge section of the Description to address these concerns.

2. The building block addresses tracking software installed to file system. Not all software is installed directly to a file system. For example, some software may be installed within a database or application server. Other installation contexts should be allowed that account for different installation contexts.

   Response: There is no reason to constrain software installation to file system-based methods. We have removed references to the "file system" and instead refer generally to the "installation environment" which allows for a number of different installation contexts to include databases, virtual containers, etc.

3. Use and meaning of the term "situational awareness" is not clear in the draft. It is not clear if this "situational awareness" is provided by humans and/or a computer system.

   Response: The text in section 1 under the "Goal" subheading has been clarified to describe the use of standardized protocols to exchange software and patch inventory data collected using specialized automation software on a device. This data can be used provide greater enterprise "situational awareness" over the software installed on computing devices as a foundational part of a continuous monitoring capability.

4. Using SWID tags to limit software execution and network access is too broad. You should consider using permission management functionality available in mobile operating systems to manage software on a much finer grained level to manage access to OS and device resources.

   Response: The goal of this building block is demonstrate the use of SWID tags, deployed during the management of software installations on devices, to support policy enforcement based on the collection of installed software inventory and software integrity measurements. Use of fine-grained application permissions for further policy enforcement is beyond the scope of this building block. This may be addressed by another project in the future.

5. It is not clear how listings and hashes of files within a SWID tag support verification of both software media pre-installation and installed software post-installation.

   Response: Changes have been made to introduce terminology and concepts in the third and fourth paragraphs of section 3. Approach relating to the use of file listings and hashes in SWID tags to support pre-installation verification of installation media and post-installation verification of installed software. These capabilities 2 and 3 amplify this approach.

6. In some installation environments, software is installed on a network share or removable drive. How will this building block address this type of installation environment?

   Response: Use of dynamically mounted drives is an area that we would like to explore under this building block. Text has been added to the 11th paragraph of section 3. Approach to clarify this intent.

7. It is not a good practice to use execution whitelisting when booting an OS in a maintenance mode such as Windows “Safe-Mode” or UNIX single-user mode.

   Response: Added text to the end of the 4th paragraph of section 3. Approach indicating that the application of whitelisting needs to be done with caution to avoid this situation. As part of the engineering work involved in developing a demonstration of this building block, we will need to consider how best to apply whitelisting capabilities to avoid preventing operation system booting/startup. To do this the capabilities of each target platform will need to be considered.

8. If the software creator’s SWID tag does not contain the full component list (e.g., libraries, executables) in the footprint, it may not be possible to whitelist software execution for that software. Use of 3rd-party SWID tags would be needed to ensure full coverage of all software components and patches. At execution this creates a potential race condition between the whitelisting capability and any 3rd-party functionality that might be deploying tags. How will this situation be handled?

   Response: The whitelisting capability will only be able to whitelist execution based on available information. Use of 3rd-party tags to address information gaps is something we would like to explore in the building block. In doing so there will be a number of “race conditions” and deconfliction scenarios that will need to be explored and addressed with regards to 1st-party and 3rd-party SWID tags.

9. Since the SWID standard only supports one of each of the footprint sections in a single tag, and it is recommended that the software creator self-heal the footprint sections, it is not advisable for Third Parties to modify the footprint sections of the software creator’s tag.

   Response: The ISO/IEC 19770-2 standard is currently undergoing revision. The 2009 version of this specification allowed for signing parts of the SWID tag to validate the integrity of the tag’s content to detect changes. The revision requires that SWID tags produced by software creators, publishers, etc. is not modified once produced. One way of addressing this revised requirement is for a supplementary tag to be created by 3rd-parties to provide additional information without changing the original tag.

10. It would be advisable to define a best practice of maintaining “base-line” tags that would define the “authorized baselines” for an endpoint. These baselines would represent a definition of what software is authorized for use on the device. These tags would have the secondary or related footprint sections populated with the list of files that are included in the package. File hashes would be omitted in these tags since they are included in the software creator’s SWID tag.

    Response: Using SWID tags for establishing software baselines is an interesting idea. Software baseline information could be used to extend both endpoint- and network-based policy enforcement capabilities. Exploration of software baseline capabilities is currently beyond the scope of this building block, but may be addressed by this or another project in the future.

11. This project should promote SAM capabilities for use in web application environments. SWID tags can be used for commercially available and custom web applications.

    Response: Addressing web application deployment environments, along with database and other compositional installation contexts, is a stretch goal of this building block. While this type of SAM capability is in scope, such functionality will likely not get addressed in the initial iterations of this building block and may be deferred to another project.

12. It would be good to tie the building block to the NIST cybersecurity framework and CAESARS-FE documents. By tying in these concepts, the building block should make clear what SAM capabilities are significantly inhibited by the lack of standardized SAM capabilities and information. It should be very clear that this SAM building block intends to demonstrate improvements to SAM capabilities based on standardized COTS implementations.

    Response: TODO: reference the controls information.

13. There are multiple standards used as part of the building block – SWID and TNC ECP. It appears that the two are linked/dependent and both must be adopted by tools for the value of SWIDs to be realized.

    Response: Text has been added to the end of the 8th paragraph of section 3. Approach to indicate that both standards are needed for this building block. All capabilities require the availability and use of SWID tags. Capabilities 2 and 3 do not require a transport protocol since no information needs to be exchanged with a peer outside the endpoint. Capabilities 1 and 4 require the use of the TNC ECP for transporting software inventory data.

14. Publishing software with standardized, high-quality SWID tags and having SAM tools capable of using these tags provides a basis for software identification and management under this building block. This represents a clear improvement over current SAM capabilities based on other proprietary and standardized approaches. A clear milestone-oriented plan is needed to communicate what is needed to drive definitive procurement requirements for SWID tags.

    Response: The purpose of this building block is to demonstrate the operational viability of using SWID tags and related standards to address a number of security challenges (see section “Security Challenge”) by realizing a number of security characteristics (see section 3). Through the production of a reference design, an associated build, and a resulting solutions guide, we hope to accelerate the adoption of commercial solutions based on this building block. Developing an implementation plan and procurement requirements for use of SWID tags is outside the mission of NIST and the NCCoE, and is beyond the scope of this project.

15. This building block should be based on clearly defined use cases that align with pressing problems resulting from poor SAM capabilities and data. The building block

should first clearly demonstrate the current challenges with multiple SAM tools (e.g., lack of standardized information and techniques, lack of integration, etc.) and then measure how the use of SWID tags resolve these issues in other capabilities.

Response: This building block identifies a number of capabilities in section 4 which roughly equate to use cases. Evaluating the capabilities of existing solutions is beyond the scope of this building block. Through the development of this building block, the security characteristics and capabilities address will be identified and documented. As indicated in the beginning of the 10th paragraph of Section 3. Approach, any gaps will be identified and any feedback will provided to the appropriate organizations.

16. As part of establishing the software environment for capability 0, a base environment needs to be established with a set of core applications across a variety of platforms (e.g., typical laptop, server, virtual) using a commonly used set of software.

    Response: The actual platforms, environments, and software used as part of this building block will be selected in cooperation with and provided by the vendors participating in the development of the build and through available open source solutions.

17. For capability 0, the technologies used for securing SWID tags on a given platform should not require new capabilities for current operating systems.

    Response: While it would be ideal to use existing access control and other technologies to secure the stored SWID tags, existing approaches may not be sufficient. We plan to explore this issue during the reference design and build processes to evaluate the use of existing approaches. Any gaps will be identified and potential mitigations will be explored.

18. This building block should validate that the ISO SWID standard meets the requirements for DHS's CDM project. It should also validate best practices outlined by TagVault.org.

    Response: This building block addresses basic secure software asset management capabilities that are needed by most enterprise environments including government agency environments. We have consulted DHS in the development of this building block and have worked to align the capabilities explored with their functional needs for continuous monitoring of software assets. As part of the reference design and build, we plan to use any appropriate best-practices for design, use of SWID tags, and implementation. Specific practices will be identified collaboratively with the organizations participating in this process and through public comment. While validation of specific requirements and best-practices is out of scope for this effort, we will document the overall approach and any best practices used and will work to identify any gaps in the existing guidance.

19. Capability 1 should be more focused on the downstream uses of exchanged software inventory data collected from endpoints. This should include use of a configuration management database (CMDB) to allow for storage and retrieval of previously exchanged data.

    Response: As part of producing a demonstration of capability 1 functionality, the NCCoE will need to identify uses of the exchanged software inventory data. This will be an active area of engineering as part of development of the reference architecture with the participating partners.

20. Regarding capability 1, there are current techniques for exchanging software inventory data. This building block should focus on normalized, standard information exchanged via SWIDs rather than focus on a new protocol. Use of the TNC protocols should be a much later capability.

    Response: One of the purposes of this building block is to demonstrate an interoperable, standards-based, platform-neutral approach for exchanging software inventory data. To achieve this degree of interoperability, we need to consider standardized transport protocols and data formats. The TNC ECP supports interoperability by providing both a standardized transport and a standardized data format with existing adoption in the marketplace. Use of these standards does not preclude the use of other standards or proprietary solutions in other deployment scenarios.

21. In capability 3, execution authority appears to be a more advanced used case. Some caution should be exercised to avoid making SWID tags appear more complicated than they actually are or that industry needs to wait until the this building block explores all of these capabilities. It needs to be clear that this building block wants to validate the most basic capabilities first, with the aim for getting the industry moving to integrate these capabilities into their available solutions quickly.

    Response: Development of this building block will be based on an iterative approach. Basic capabilities will be explored in capabilities 0 and 1. Capabilities 2, 3 and 4 represent advanced building blocks for SWID tags that are included as stretch goals. For each build iteration, we will collaborate with the build participants to determine what capabilities to incorporate. Based our initial analysis, we believe there are existing fielded APIs and capabilities that provide the pieces needed to fully explore this building block. Some minimal "glue code" may be needed to integrate these capabilities as part of developing this building block. What will not be clear until we get further into the reference design and build process is how much glue code will be needed to knit these capabilities together.