CDM Software Asset Management (SWAM) Capability

Department of Homeland Security
Office of Cybersecurity and Communications
Federal Network Resilience

Table of Contents

1 PURPOSE AND SCOPE

This tool outlines and documents issues of relevance to implementing the Software Asset Management (SWAM) Capability as part of Continuous Diagnostics and Mitigation (CDM). This toolkit provides general information on the SWAM capabilities and implications thereof. Further, this toolkit highlights consideration that technical implementers as well as managers may have when understanding how to effectively implement SWAM to better manage cybersecurity risk.

Additional considerations, inquiries, and suggestions for revision or addition can be submitted to: cdm.fnr@hq.dhs.gov. This toolkit will be updated as required.

2 THREAT / ATTACKS

1. QUESTION: WHAT TYPES OF ATTACKS ARE WE TRYING TO ADDRESS WITH SOFTWARE ASSET MANAGEMENT (SWAM)?

Answer: SWAM addresses attacks that result from unauthorized software and from malicious software. SWAM also indirectly addresses attacks based on unsafe configurations and out-of-date patches.

Background: Because modern software is complex and consists (typically) of thousands of files per product, it is very difficult for software managers to ensure that such software is:

- authorized & needed
- from a trusted supply chain (as opposed to including malicious code placed there by malefactors)
- safely configured (to reduce the attractiveness of particular targets and the likelihood of successful attacks)
- using current patches (to reduce the attractiveness of particular targets and the likelihood of successful attacks)

As a result, attackers frequently compromise devices by finding and exploiting weaknesses. SWAM addresses attack scenarios resulting from the difficulties related to the items above, particularly the first two bullets. SWAM indirectly supports the second two items. The quality of configuration settings (CCEs) and patching (CVEs/CWEs) for authorized software (with assigned setting/patch managers) are covered under the Configuration Setting Management (CSM) and Vulnerability Management (VUL) capabilities, respectively. SWAM supports the CSM and VUL capabilities by making sure we know what software is present so that it can be correctly configured and patched.

*The Software Asset Management capability addresses attacks resulting from **lack** of management of configuration settings or system vulnerabilities; it does **not** address attacks resulting from **poor** management of configuration settings or system vulnerabilities. Poor management by those assigned to address settings and patches is handled by CSM and VUL respectively.*

*The Software Asset Management capability does not address attacks on devices that are **unmanaged**. Attacks on unmanaged devices are handled by the Hardware Asset Management capability.*

If a device is managed *under hardware asset management, we assume that it is the responsibility of the device manager to either manage the software on* *the device, or make sure that the software is managed. The roles involved in managing the software that might be delegated by the device manager are discussed below.*

2. QUESTION: HOW DOES SOFTWARE ASSET MANAGEMENT ADDRESS THESE TYPES OF ATTACKS?

Answer: This capability addresses attacks that exploit the weaknesses listed in Question 1, as follows:

Weakness	**How Addressed**
Unauthorized Software Products	• Directly o Assign devices to role(s). o For each device role, know what software products are authorized. o Block other products from running, remove them, or accept the risk (through a device-specific authorization).
Unauthorized Executables • Malware inserted as unauthorized executables • Malware inserted into authorized executables	• Directly o For authorized products, know the executable files associated with each. Validate file integrity through a known digital fingerprint for each executable file. o Block files that don't have a correct fingerprint associated with an authorized product.
Unmanaged settings and patches (CCEs and CVEs/CWEs)[1]	• Indirectly o Ensure that a manager with the appropriate skills and privileges is assigned to manage these factors on each installed instance of a product. o If no such manager(s) are explicitly assigned, assume that the device manager is responsible. o Report all defects to the assigned manager for correction and/or risk acceptance.

[1] Once software is identified, CCEs are directly handled by Configuration Setting Management, and CVEs/CWEs are directly handled by Vulnerability Management.

How hard is it to address weaknesses using the methods listed? *There are millions of products–each product with thousands of executables—on many government networks. The fingerprints for each executable may change with each patch. Performing the tasks to address weaknesses manually would be infeasible.* ***Appropriate automation is required.*** *The automation tool must have data on the executables associated with each product and the valid fingerprints for each executable. It must also be able to selectively block products and executables. If the tool also provides data on known bad products and executables it further simplifies the process.*

3. QUESTION: WHAT IS A SOFTWARE ASSET?

Answer: In general, a software asset can be as small as a line of source code or as large as a software suite made up of multiple products, thousands of individual executables, and countless lines of code.

Background: Software includes firmware, basic input/output systems (BIOS), operating systems, applications, services, and malware such as rootkits, trojans, viruses, and worms. Software assets are often described in terms of Common Platform Enumeration (CPE). CPE is a standardized method of describing and identifying classes of applications and operating systems present among an enterprise's computing assets.

Products and Executables: In order to manage software in an efficient way, we must select the level of assets to be managed. In CDM, we focus on *products* and *executables:*

a) **PRODUCTS:** It is important to manage software *products* because this is the level of abstraction by which software is typically licensed, listed in registries during installation, and executed by users. By *product* we mean a specific combination:
- Vendor
- Product
- Release
- Patch Level

Are suites products? Suites are typically collections of products (as defined above) licensed and sold together. In cases where the individual products have the same content (registry entries, executables, etc.) whether sold individually or in the suite, it is simpler to ignore the suite, and focus on the products.

Software products are roughly equivalent to the software identified by the NIST Common Product Enumeration (CPE) (http://nvd.nist.gov/cpe.cfm) codes, and also by the ISO SWIDs (http://www.iso.org/iso/catalogue_detail.htm?csnumber=53670).

b) **EXECUTABLES:** To eliminate malware, it is important to focus on executable files. Malware may be introduced as a single file not associated with a product, or it may be introduced by substituting a file with malicious capability for an authorized executable within an authorized product. Linking executables to products and being able to validate that the executable present is the same one sent by the vendor are vital to identifying malware. Thus, an executable defined as "a specific file in persistent memory that can be loaded into active memory and executed by the CPU" is a software asset.

At present, SWAM does not include mobile code that is downloaded each time it is executed as a software asset since network managers cannot generally configure or patch such code. Moreover, there is no good source for verifying that the code received matches what was intended. However, as the ability to validate whether such code should be trusted (or not) improves, mobile code could and should be treated as an asset to be managed.

In the case of interpreted languages, source code may be treated as an executable even though it will be interpreted before being loaded in memory. The digital fingerprint of the source code file can be used to verify that it is the intended code. However, if the source code is compiled into machine code and only the machine code is loaded on the device where the product runs, the compiled code should be the only asset found on the target device.

ISO software identification tags (SWIDs) are capable of recording and reporting the list of executables that make up the product and the digital fingerprints of each file. This capability supports two meaningful supply chain analyses: First, do the executables reported in the SWID match the expected executables for the product? If not, the product may have been tampered with before installation. Second, do the executables reported in *the SWID match the executables currently on the device? If not, the product may have been tampered with after installation.*

Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE can be used as a source of information for enforcing and verifying IT management policies relating to these assets, such as vulnerability, configuration, and remediation policies. (See https://register.mitre.org/devdays/certified_swid_tags_integration_cpe_names.pdf.)

TagVault.org is a registration and certification organization which provides tools that are used internally within a software publisher's environment to validate normalized and registered names, to validate that a minimum amount of data is included in the software identification (SWID) tag based on certification level, and to digitally sign the certified tag. (See https://register.mitre.org/devdays/certified_swid_tags_integration_cpe_names.pdf.)

Links to Related Resources
• Certified SWID Tag Integration with Common Platform Enumeration names https://register.mitre.org/devdays/certified_swid_tags_integration_cpe_names.pdf
• ISO/IEC 19770-1:2012 -- Information technology -- Software asset management -- Part 1: Processes and tiered assessment of conformance http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=56000
• ISO/IEC 19770-2:2009 -- Information technology -- Software asset management -- Part 2: Software identification tag http://www.iso.org/iso/catalogue_detail.htm?csnumber=53670

3 INTEGRATION

4. QUESTION: WHAT CAPABILITIES SUPPORT SOFTWARE ASSET MANAGEMENT?

Answer: Hardware Asset Management (HWAM).

Hardware Asset Management is the only capability that directly supports the implementation of software asset management. It is impossible to check for a software asset if we do not know where it was or should be installed.

You can't manage (or find) software on a device if you don't know the device is there.

5. QUESTION: WHAT CAPABILITIES DOES SOFTWARE ASSET MANAGEMENT SUPPORT?

Answer: Vulnerability management and configuration settings management.

Software asset management is necessary for proper execution of both vulnerability management and configuration settings management: an organization must complete software asset management before it can implement vulnerability management and configuration settings management. In other words, it is impossible to check for software versions or settings under configuration settings management or the status of patches under vulnerability management without first having an inventory of software within an organization.

You can't manage (or find) needed configurations or patches for a software product if you don't know the product is there.

6. QUESTION: WHAT CAPABILITIES PROVIDE "COMPENSATING CONTROLS" TO SOFTWARE ASSET MANAGEMENT?

Answer: Vulnerability management and configuration settings management.

Both vulnerability management and configuration settings management provide compensating controls to software asset management as both capabilities, when properly implemented, have the potential to discover unauthorized or unknown software installed in the network.

Many tools can help identify what software is in the actual state.

4 HOW DOES SOFTWARE ASSET MANAGEMENT PREVENT ATTACKS FROM “BAD” SOFTWARE ASSETS?

7. QUESTION: HOW CAN MALICIOUS SOFTWARE BE MANAGED THROUGH BLACKLISTS AND WHITELISTS?

Answer: From a trust perspective, software can be categorized according to the following lists:

- White list - “known good” or trusted software
- Black list - “known bad” or untrusted software
- Gray list - observed software that is not known to be good or bad at a moment in time

Whitelisting (blocking all software except for trusted software) and blacklisting (blocking all untrusted software) can be used as strategies to block malware.

Should unauthorized software be kept off a device (whitelisting) and/or have its execution blocked (blacklisting)? *Each of these approaches can be used alone to keep untrusted software from running. However, using both approaches together provides compensating controls so that if one fails, the other can block execution. In particular, whitelisting software that blocks execution of malware that may be downloaded inadvertently is a more effective protection against such malware.*

Whether to allow graylisted software to run is a judgment call. See Question 8.

8. QUESTION: IS ALL GRAYLISTED SOFTWARE THE SAME?

Answer: No. The gray list may be divided into two parts:

- Allowed gray list – List of graylisted items that we will assume are safe and allow to execute. This might include, for example, software newly installed by approved personnel at an agency which distributed a Software Configuration Control Board.
- Disallowed gray list – All other graylisted software, which should not be allowed to run. This might include software not installed/downloaded by an approved installer (for example, as a result of a phishing attack).

There is an increased risk in allowing graylisted software to run. This risk must be reflected in risk scoring. The organization should seek to minimize *the amount of graylisted software by determining the trustworthiness of the software and moving it to the white or black lists.*

9. QUESTION: SHOULD THE BLACKLIST AND WHITELIST BE APPLIED IN THE SAME MANNER FOR ALL DEVICES, REGARDLESS OF DEVICE ROLE?

Answer: No. These lists are not universal. Different kinds of devices have different roles and need different software. Examples include:

- Routers, user workstations, and mobile devices often have very different operating systems.
- Software that is appropriate on Windows servers (for example, MS-Exchange) should not be installed on user workstations (which might be allowed to have, for example, MS-Outlook).

The organization must define enough device roles so that whitelists and blacklists can be specific enough to limit software to that needed on the device without requiring a large number of exceptions.

Definition: *Device role* means the business and/or technical function(s) that the device is intended to perform, such that the role correlates closely with (and determines) the software needed on the device.

10. QUESTION: ARE WHITELISTS OR BLACKLISTS BETTER FOR BLOCKING BAD SOFTWARE?

Answer: Whitelisting has strong potential to block more malware than blacklisting.

Blacklists: Most antivirus software operates by blocking software that is known to be bad; it operates as a blacklist. Periodic antivirus updates provide data to identify new known bad signatures. Given this approach, attackers have found blacklisting relatively easy to defeat by modifying malicious code slightly to prevent detection. This has created an significant increase in the absolute number of known bad signatures as well as in the amount of malicious code that slips through antivirus (or other blacklisting) software before it becomes "known." As a result, blacklisting is increasingly defeated and ineffective.

Whitelists: An alternate model is to allow only known good software to be executed; this is a whitelisting approach. Because the organization controls its whitelist and can keep it to a certain size, the whitelist is easier to manage. The whitelist cannot be manipulated as easily by the attacker. New versions of malware are automatically blocked (when software is identified at the executable level). Thus, whitelisting has strong potential to block more malware, including advanced persistent threats and zero-day malware.

5 DESIRED STATE

11. QUESTION: WHAT IS THE DESIRED STATE FOR SOFTWARE ASSET MANAGEMENT?

Answer: The desired state for software asset management is that every authorized device contains only the authorized (known good) software required for the role the device has been assigned. In the short term, an organization may permit selected software in an allowed graylist to execute, but this is less than ideal. In this desired state:

- Only known good software (whitelisted software) and selected graylisted software (not known to be bad or good) are allowed to be installed on the device or allowed to execute.
- Information about individuals who manage software is available so risks can be reported to the responsible person.

12. QUESTION: ARE SOFTWARE MANAGEMENT FUNCTIONS PART OF THE DESIRED STATE?

Answer: Depending on the organization, responsibility for software management may differ. In simple cases, each device manager is responsible for all software management functions on that device. In complex cases, different software management functions may be assigned to various groups, which must coordinate to manage software on a device. The desired state data for each software product must specify the person responsible for each software function for that product (on each device) so that risks/defects can be reported for resolution to the responsible person.

Software Product Management Functions:

- Product Management (across the organization)
- Installation (by device)
- Configuration Setting Management (by device)
- Patch Management (by device)

It is very important to ensure that accounts that are allowed to perform installation, configuration settings, and patching do not become infected by malware that might typically be delivered via email and web browsing. While these activities are not formally part of software asset management, preventing these accounts from accessing email and browsers is vital to preventing malicious software from being installed.

The choice of who manages software may affect risk scoring rules, as they change the amount of risk managed by each group.

13. QUESTION: HOW SHOULD/COULD AN ORGANIZATION ASSIGN SOFTWARE MANAGEMENT FUNCTIONS AND ROLES?

Answer: Organizations should select a subject matter expert for the application (role or software suite) who understands the system and software dependencies, the security configurations, the patch cycle, and how to properly install the software. There are many ways to assign management tasks:

Choice 1: Automated or Manual
In many organizations, all software is installed *manually* by an administrator who runs install software on each local device. In other organizations, installation may be done *automatically* from a centralized console that uses a software installation tool to push the new software to all devices (in some scope). Both—or a combination—of these approaches are valid, but they typically change who is responsible; this change in responsibility must be reflected in the desired state data. Where an automated tool manages installation in a known scope, that scope can be used to record the automated tool's responsibility.

Choice 2: Centralized or Decentralized
Even in organizations using manual install procedures, some products (for example, database management systems) may require a central subject matter expert to perform the installation. Likewise, in patching, an organization may use automated *packages* to install software, executing the installation not from a central console, but from distributed locations. (This might be done to allow verification that the software install doesn't interfere with "local" software.) Clearly, if software is managed from a central console through full automation, the management will likely be centralized. The scope of responsibility can also change who is responsible for a specific function.

Choice 3: Generalist (device manager) or Specialist (e.g., SME in patching, or SME in the product)

The choices for how software management is performed may be different for different asset types. For example, custom software may be managed differently than COTS software.

14. QUESTION: ARE ANTIVIRUS PRODUCTS DIFFERENT FROM SOFTWARE ASSET MANAGEMENT?

Answer: No. Antivirus products are part of software asset management; they typically operate by blocking known bad software**.** Antivirus products can be classified as generic blacklisting tools that are able to scan a system for known bad software. A robust deployment of antivirus products throughout the hosts in your enterprise is one of many tools that may be used to properly implement software asset management.

Most current antivirus products are evolving towards trust by being able to perform heuristic scans of executables. Such heuristic scans can identify types of malicious programming such as the function calls or linked libraries a particular executable attempts to call, but antivirus products are still primarily blacklisting products.

15. QUESTION: WHAT IS THE *DESIRED STATE* INVENTORY FOR SOFTWARE ASSET MANAGEMENT?

Answer: The desired state inventory for SWAM includes

- a list of the device role (see Question 13) for every authorized device in the HWAM desired state
- a list for each role that identifies the software products and executables allowed to be installed/executed for each device role
- a complete listing, by device, of all authorized software that may be installed/executed on each device in the HWAM inventory

Definition: A software profile is the list of authorized software allowed to be installed and allowed to run for a device to execute the role it has been assigned.

Most agencies have some form of configuration control board able to assess and authorize software allowed on the enterprise. Most agencies implement the results of such boards through a defined set of checklists used in the creation of standard systems of predefined roles.

A problem facing most organizations is that this control board process misses a large amount of software which is actually installed and operates at the product level. Thus, it often misses drivers installed with peripheral devices, executables changed by patching, and other software products. By operating at the product level, it has little insight into executables and their supply chain.

Unfortunately, it would be almost impossible for individual organizations to manage such authorizations for all products on their network, much less all executables. This is why it is so important to have automated tools that rely on community assessment of products and executables to facilitate this process.

16. QUESTION: WHAT DATA SHOULD BE RECORDED IN DESIRED STATE?

Answer: The minimal software asset management data recorded for the desired state should include the following:

Data Item	**Justification**
Software CPE (vendor, product, version, release level) or equivalent (such as SWID) for approved software	• Reporting device types • Supply chain management • Determining what products may apply to devices
Management responsibility for each CPE management function (see Question 13)	Identifying management responsibilities for ensuring that licensing, patching, and configuration standards are up-to-date. (If not specified explicitly, this is assumed to be the device manager.)
Time period that a software product is expected to be on the whitelist/blacklist/graylist	For identifying the life span of a particular authorized software product. (All authorized software must have a "remove by" date.)
Digital fingerprints of expected software executables (components, EXEs, DLLs, etc.) for the approved software products.	To identify when the executable may have changed, meaning it may no longer be "good"

Digital fingerprints for executables (or any file) are long numbers computed by a hashing *algorithm from the contents of the file. These fingerprints make it essentially impossible to change the file contents without changing the hash. Therefore, the digital fingerprint can be used to verify that software has not been changed from one point in the supply chain to another. When fingerprints are collected closer to the origin of the supply chain, they are more authoritative and thus more likely to be useful in detecting tampering.*

By design, this is a minimal list of data to be used as a starting point. There are many operational and security reasons that more data may be required.

If you collect other data you think is essential and/or useful, please document it and let us know so we can report your ideas here for other organizations to consider.

17. QUESTION: HOW DOES AN ORGANIZATION DETERMINE ITS DESIRED STATE?

Answer: Desired state is an organization's authorized software inventory—including asset characteristics—that is derived from the organization's processes and procedures. (See Question 13].

The authorized software inventory includes

- Establishing and maintaining a list of roles and associated profiles of authorized software products and executables permitted on the authorized hardware device with that role. This may include
    - Whitelisted software (see Question 7)
    - Allowed graylisted software (see Question 8)
- Recording exceptions where a specific device may have additional software (risk acceptance).
- Establishing and maintaining a list of software packages, products, and files that are explicitly disallowed on the network (blacklisted). These can range from packages that should not be installed on a corporate network due to their nature (e.g., pornography, gambling), to obsolete and vulnerable versions of corporate software (e.g., old versions of COTS), to invasive software packages that install unnecessary features on the network (e.g., toolbars, spyware, key loggers). Note: The blacklist is unnecessary if only whitelisted software is required to execute; however, an added blacklist does provide defense in depth.

Many whitelisting tools identify software with what the industry considers to be known good and known bad signatures. This can help identify trusted software and avoid putting known bad signatures on the allowed graylist.

Typically, whitelisting products provide digital fingerprints for executables in most commercial software products. This can help identify software executables that may have been modified to include malware. For custom code, an organization should perform CWE analysis on the code and compute its own digital fingerprints on validated code.

CWE analysis is the process of finding common weaknesses in computer code. It is an open process, and many current coding weaknesses are found and documented by the CWE initiative. (See http://cwe.mitre.org/data/index.html.)

18. QUESTION: HOW DOES AN ORGANIZATION DETERMINE WHEN CHANGES NEED TO BE MADE TO THE DESIRED STATE?

Answer: There are many events that should trigger changes to the desired state. An organization must identify and respond to such events in a timely manner. Examples follow:

Event	**Response**
A new software product or executable was installed by an approved installer.	This could be automatically allowed to run or be blocked. Typically, unless already classified as a known "good," it would be added to the graylist. Because the graylist should be small (and should get smaller over time), software products should be evaluated and moved to the white or blacklists quickly.
Legacy software that is in the allowed graylist exists on devices.	Over 12-24 months, such software must be either whitelisted or blacklisted.
Unidentified software appears on a device that was not put there by an authorized installer, or the digital fingerprint of a formerly trusted executable changes for no apparent reason.	This is likely to be malware. The organization needs a way to prevent such software from executing (because it is not whitelisted); the organization must also have a way to remove it and determine its origin.
Digital fingerprints of expected software executables (e.g., components, EXEs, DLLs) for the approved software products have changed.	The organization must use digital fingerprints to identify when the executable may have changed, meaning it may no longer be "good." Changed executables may need to be removed.

6 ACTUAL STATE

19. QUESTION: WHAT IS ACTUAL STATE?

Answer: The actual state for software asset management is 1) every software product and 2) every executable file that currently exists on enterprise networks, including correlations of the executable files with their corresponding products. This includes:

- Authorized and unauthorized software
- Malicious and non-malicious software
- Software allowed to run and software not allowed to run

It would be desirable to include mobile code, as well. However, it is currently infeasible to get validated authoritative digital fingerprints to verify mobile code.

20. QUESTION: WHAT MAKES UP THE ACTUAL STATE INVENTORY?

Answer: The actual state inventory for software asset management is a listing, by discovered device, showing all discovered software (products and corresponding executables) found to exist on each device in the organization.

It is essential that this list of discovered software be complete and include all the software in the actual state, to the maximum extent possible.

21. QUESTION: WHAT DATA SHOULD BE RECORDED IN ACTUAL STATE?

Answer: The minimal software asset management data recorded for the actual state should include:

Data Item	**Justification**
Software assets (products and executables) detected on a hardware asset	For future comparison of white and blacklisted software
Time period the software (CPE/executable) is estimated to have been present	To determine how long the software has been seen and the last time the software was seen in the enterprise
Sufficient data to allow comparison of the actual software asset to authorized hardware assets	For linking authorized software to the devices it should be installed on • Mapping to CPE, SWID, etc. for products • Digital fingerprint for executables
Data to uniquely identify the correct content of each digital object that makes up the software CPE (comparison to the digital fingerprint)	For correlating each unique identifier to each executable that makes up a software suite
The data necessary to deduce the software CPE (or equivalent, such as SWID) from the collection of software assets (identified digital objects)	For linking CPEs (or equivalent, such as SWID) to a particular software asset

By design, this is a minimal list of data to be used as a starting point. There are many operational and security reasons that more data may be required.

If you collect other data you think is essential and/or useful, please document it and let us know so we can report your ideas here for other organizations to consider.

22. QUESTION: HOW DOES AN ORGANIZATION DETERMINE ITS ACTUAL STATE?

Answer: The CDM program has established a strategically-sourced solution to provide sensors to collect data about the software that exists throughout your network. These sensors perform periodic software discovery and logging through a range of techniques such as registry scans, system self-reporting, and individual file analysis.

As of 2013, most agencies have some form of configuration control board able to assess and authorize software allowed on enterprise. Most agencies implement the results of such boards through a defined set of checklists used in the creation of standard systems of predefined roles. The problem facing most organizations is a lack of insight into the actual state of their systems, especially with respect to being able to determine whether a graylist item should be authorized or classified as malicious. Without both desired and actual inventories, it is not feasible to identify missing or unauthorized software.

7 FINDING RISK CONDITIONS AND DEFECTS

23. QUESTION: HOW DOES AN ORGANIZATION FIND THE DIFFERENCE BETWEEN DESIRED STATE AND ACTUAL STATE?

Answer: If the desired state inventory and actual state inventory are set up correctly, all the data necessary to detect these potential risk conditions is available, and the risks can be detected with a simple database query. The results of the query can then be reported to a dashboard which can record when each risk condition was found and how long it existed. **Under CDM, this process is automated and handled by the CDM dashboard after data is received from the desired state inventory and the actual state sensors.**

Definition: A difference between the desired state and the actual state is considered to be a risk condition that can cause defects. The primary examples are listed in the table below:

Difference detected between desired and actual state	**Why is this considered a risk condition?**	**the Whatrisk? What is the likely risk?**
Blacklisted software is allowed to be present or execute.	Known bad software typically causes security compromise.	The software is likely to cause a security compromise.

Graylisted software is allowed to be present or to execute.	Software not verified to be safe is likely to be executed.	Because the software has not been verified to be safe, there is an elevated risk that it will cause a security compromise.
Unauthorized software is present.	Software not listed in the desired state is allowed to be present or to execute.	Because the software has not been verified to be safe, there is an elevated risk that it will cause a security compromise. Because the software does not appear in the desired state list, there is an elevated risk that it will cause a security compromise.
Non-reporting devices are present.	Because the device is not reporting what software is present, there is an elevated risk of malware being present.	Typically, if a device is not reporting, the software manager(s) don’t know what needs to be fixed, so defects increase over time. There is also a risk that the device may not be reporting because malware or a malicious insider has interfered with reporting.

8 FIXING DEFECTS

24. QUESTION: WHAT OPTIONS ARE AVAILABLE FOR ADDRESSING THE DIFFERENCES BETWEEN ACTUAL AND DESIRED STATE?

Answer: Once the CDM dashboard has identified the differences between actual and desired state inventories, the dashboard will mark the software assets for which there are risk conditions. Scoring algorithms will be applied to estimate the risk. The risk score may increase if the risk condition is older (over time, the probability of compromise increases). The risk score may also be higher for some kinds of software.

Actual removal of the risk will require the following actions:

Risk Condition	Detection Rule	Response Options
Blacklisted software is allowed to execute.	Software appears in the actual state inventory but is blacklisted in the desired state inventory, and execution is not blocked.	1. Block the software from executing. 2. Remove it from the actual state (uninstall). 3. Un-blacklist it. (This response is dangerous.)
Graylisted software is allowed to execute.	Software appears in the actual state inventory but is graylisted in the desired state inventory, and execution is not blocked.	1. Block the software from executing. 2. Remove it from the actual state (uninstall). 3. Whitelist it. (This response could be dangerous.) Issue: This defect starts as a lower risk than the previous one, but the risk can increase over time.
Unauthorized software is present.	Software appears in the actual state inventory but not in the desired state. (Note: An organization may automatically graylist software if it is installed by an approved installer to avoid this problem.)	Add the software to the desired state, and white/gray/blacklist it.

Non-reporting devices are present.	The hardware which has the software is in the desired or actual state inventory, but not in the software actual state with timely-enough data.	Restore reporting, or declare the device missing/uninstalled/retired in HWAM.

25. QUESTION: HOW CAN WE PREVENT UNAUTHORIZED SOFTWARE FROM GETTING ON THE NETWORK IN THE FIRST PLACE?

Background: Removing unauthorized software or authorizing software from the graylist is something that theoretically shouldn't be necessary. But in real operational networks, it often seems inevitable and in the end is a daunting task. Still, one must ask if there are things the organization could do to reduce the rework required to find and classify newly discovered software as authorized or malicious.

Answer: The following kinds of actions can be taken to *reduce* the amount of unauthorized software on systems throughout your enterprise:

1) **Policy can state who will have elevated privileges on systems and what they can do with said privileges.** System administrators often introduce unauthorized and potentially malicious software to systems because they are able to install new software on systems and use the internet with elevated accounts, or they may use poor sanitary practices with respect to removable media. Properly limiting who can hold such elevated privileges and defining authorized and unauthorized activity will greatly limit your exposure to future risk.

2) **Logging can track when unauthorized software is installed and by whom.** Such logs can assist in determining root cause (e.g., breakdown in configuration management process, abuse of privileges). Once the person is found, letting them know what is expected can prevent creation of these risk conditions.

3) **A few people may need to be sanctioned.** Sanction individuals who frequently install unauthorized software through the abuse of their role and privileges, and who do so after due warning.

While such actions won't eliminate all unauthorized software, these actions can lower their incidence rates, which is a positive step.