The attached DRAFT document (provided here for historical purposes) has been superseded by the following publication:

Publication Number: **NIST Special Publication (SP) 800-167**

Title: **Guide to Application Whitelisting**

Publication Date: **10/30/2015**

- Final Publication: http://dx.doi.org/10.6028/NIST.SP.800-167 (which links to http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-167.pdf).
- Related Information on CSRC: http://csrc.nist.gov/publications/PubsSPs.html#800-167
- Information on other NIST Computer Security Division publications and programs can be found at: http://csrc.nist.gov/

The following information was posted with the attached DRAFT document:

Aug. 22, 2014

SP 800-167

DRAFT Guide to Application Whitelisting

NIST announces the public comment release of Draft Special Publication (SP) 800-167, Guide to Application Whitelisting. The purpose of this publication is to assist organizations in understanding the basics of application whitelisting (also known as application control) by examining the basics of application whitelisting and explaining the planning and implementation for application whitelisting technologies throughout the security deployment lifecycle.

Please send your comments to 800-167comments@nist.gov by September 26, 2014 using the following template

NIST Special Publication 800-167 (Draft)

Guide to Application Whitelisting

Adam Sedgewick
Murugiah Souppaya
Karen Scarfone

C O M P U T E R S E C U R I T Y

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

NIST Special Publication 800-167 (Draft)

Guide to Application Whitelisting

Adam Sedgewick
Information Technology Laboratory

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Karen Scarfone
Scarfone Cybersecurity
Clifton, Virginia

August 2014

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Acting Under Secretary of Commerce for Standards and Technology and Acting Director

Authority

This publication has been developed by NIST to further its statutory responsibilities under the Federal Information Security Management Act (FISMA), Public Law (P.L.) 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-167
Natl. Inst. Stand. Technol. Spec. Publ. 800-167, 26 pages (August 2014)
CODEN: NSPUE2

Public comment period: *August 25, 2014* through *September 26, 2014*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Email: 800-167comments@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

An application whitelist is a list of applications and application components that are authorized to be used in an organization. Application whitelisting technologies use whitelists to control which applications are permitted to execute on a host. This helps to stop the execution of malware, unlicensed software, and other unauthorized software. This publication is intended to assist organizations in understanding the basics of application whitelisting. It also explains planning and implementation for whitelisting technologies throughout the security deployment lifecycle.

Keywords

access control; application control; application whitelisting; information security; whitelisting

Acknowledgments

The authors, Adam Sedgewick and Murugiah Souppaya of the National Institute of Standards and Technology (NIST) and Karen Scarfone of Scarfone Cybersecurity, wish to thank their colleagues who contributed to this publication.

Trademark Information

Table of Contents

List of Appendices

Executive Summary

An *application whitelist* is a list of applications and application components (libraries, configuration files, etc.) that are authorized to be present or active on a system according to a well-defined baseline. The technologies used to apply application whitelists—to control which applications are permitted to install or execute on a host—are called *whitelisting programs, application control programs,* or *application whitelisting technologies*. Application whitelisting technologies are intended to stop the execution of malware and other unauthorized software. Unlike security technologies such as antivirus software, which block known bad activity and permit all other, application whitelisting technologies are designed to permit known good activity and block all other. The purpose of this publication is to assist organizations in understanding the basics of application whitelisting and planning for its implementation.

Implementing the following recommendations should facilitate more efficient and effective application whitelisting use for Federal departments and agencies.

Consider using application whitelisting technologies already built into the host operating system.

Organizations should consider these technologies, particularly for centrally managed hosts, because of the relative ease in managing these solutions and the minimal additional cost. If built-in application whitelisting capabilities are not available or are determined to be unsuitable, then the alternative is to examine third-party solutions with robust centralized management capabilities.

Use products that support more sophisticated application whitelisting attributes.

Choosing attributes is largely a matter of achieving the right balance of security, maintainability, and usability. Simpler attributes such as file path, filename, and file size should not be used by themselves unless there are strict access controls in place to tightly restrict file activity, and even then there are often significant benefits to pairing them with other attributes. A combination of digital signature/publisher and cryptographic hash techniques generally provides the most accurate and comprehensive application whitelisting capability, but usability and maintainability requirements can put significant burdens on the organization.

Test prospective application whitelisting technology in monitoring mode.

It is highly recommended to test any prospective application whitelisting technology in a monitoring mode to see how it would behave before solution deployment. This testing should include a thorough evaluation of how the solution reacts to changes in software, such as installing an update. An application whitelisting technology might be considered unsuitable if, for instance, it had to be disabled in order to install security updates for the operating system or particular applications.

Address application whitelisting technology planning and deployment in a phased approach.

A successful deployment will require a clear, step-by-step planning and implementation process. The use of a phased approach for deployment can minimize unforeseen issues and identify potential pitfalls early in the process. This model also allows for incorporating advances in new technology and adapting the technology to the ever-changing enterprise. In addition to following the security recommendations presented in this publication, organizations implementing application whitelisting technologies should also follow the recommendations from NIST Special Publication (SP) 800-53, *Recommended Security Controls for Federal Information Systems and Organizations*, which defines minimum recommended management, operational, and technical controls for information systems based on impact categories.

When evaluating the possibility of deploying application whitelisting, analyze the environment or environments in which the application whitelisting will be running.

It is more practical to implement whitelisting on hosts that are centrally managed and have a consistent application workload. Application whitelisting solutions are generally strongly recommended for hosts in high-risk environments where security outweighs unrestricted functionality. Suitability for typical managed environments depends on how tightly the hosts are managed and the extent of the risks that they face. Organizations considering application whitelisting deployment in a typical managed environment should perform a risk assessment to determine whether the security benefits provided by application whitelisting outweigh its possible negative impact on operations. Organizations should also be mindful that they will need dedicated staff managing and maintaining the application whitelisting solution depending on the scale and specifics of the solution implemented, similar to handling an enterprise antivirus or intrusion detection solution. An organization that can dedicate the necessary trained staff to solution maintenance and has built-in application whitelisting technology should generally implement application whitelisting at least in a monitoring mode.

1. Introduction

1.1 Purpose and Scope

The purpose of this publication is to assist organizations in understanding the basics of application whitelisting (also known as application control). All other forms of whitelisting, such as email, network traffic, and mobile code whitelisting, are out of the scope of this publication.

1.2 Audience

This document is intended for security managers, engineers, administrators, and others who are responsible for acquiring, testing, implementing, and maintaining application whitelisting technologies.

1.3 Document Structure

The remainder of this document is organized into the following sections and appendices:

- Section 2 examines the basics of application whitelisting.
- Section 3 explains planning and implementation for application whitelisting technologies throughout the security deployment lifecycle.
- Appendix A provides a mapping to existing standards and guidelines that support using application whitelisting technologies.
- Appendix B contains a detailed discussion of a particular problem that is relevant across a variety of industry sectors.
- Appendix C discusses considerations involved in applying application whitelisting technologies to mobile platforms.
- Appendix D defines selected acronyms and abbreviations used in the document.
- Appendix E lists the references for the publication.

2. The Basics of Application Whitelisting

A *whitelist* is a list of discrete entities, such as hosts, email addresses, network port numbers, runtime processes, or applications that are authorized to be present or active on a system according to a well-defined baseline. A *blacklist* is a list of discrete entities that have been previously determined to be associated with malicious activity. A *graylist* is a list of discrete entities that have not yet been established as benign or malicious; more information is needed to move graylist items onto a whitelist or a blacklist. Whitelists, blacklists, and graylists are primarily used as a form of access control: permitting activity corresponding to the whitelist and not permitting activity corresponding to the blacklist. Graylist treatment depends on the type of entities it contains, but an example of how a graylist might be handled is prompting the user to make a decision or notifying an administrator that the entity needs to have its security evaluated before use.

An *application whitelist* is a whitelist of applications and application components (libraries, configuration files, etc.) The technologies used to enforce application whitelists—to control which applications are permitted to be installed or executed on a host—are called *whitelisting programs, application control programs,* or *application whitelisting technologies*. Application whitelisting technologies are intended to stop the execution of malware and other unauthorized software. Unlike security technologies such as antivirus software, which use blacklists to block known bad activity and permit all other, application whitelisting technologies are designed to permit known good activity and block all other.

This section examines the basics of application whitelisting. It first discusses the categories of threats that application whitelisting can mitigate and the types of application whitelisting. Next, it defines the types of operational runtime modes available for application whitelisting technologies. The section also explains the motivations for application whitelisting and discusses uses of application whitelisting technologies other than application access control. Finally, the section concludes by examining differences in deployment based on operational environment, as well as considerations for evaluating the relative effectiveness of application whitelisting solutions.

2.1 Threats

As previously discussed, application whitelisting software prevents installation and/or execution of any application that is not specifically authorized for use on a particular host. This mitigates multiple categories of threats, including malware and other unauthorized software.

Malware, also known as malicious code, refers to an application that is covertly inserted into another piece of software (e.g., operating system, application) with the intent to steal or destroy data, run destructive or intrusive programs, or otherwise compromise the confidentiality, integrity, or availability of the victim's data, applications, or operating system.[1] Many of today's threats are malware-based, attempting to infect hosts (install their malicious code) and execute on those hosts to steal their data or perform other harmful activities. When properly configured, application whitelisting technologies can stop most malware from being executed (and often from being installed in the first place). Application whitelisting technologies can be significantly more effective at stopping unknown malware threats than conventional antivirus software and other traditional antimalware security controls. This is important because today's malware threats are increasingly customized and targeted, making traditional detection technologies largely ineffective.

[1] This definition is based on the one provided in NIST Special Publication 800-83 Revision 1, *Guide to Malware Incident Prevention and Handling for Desktops and Laptops* (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-83r1.pdf).

The other major category of threats that application whitelisting technology can mitigate is other unauthorized software (unauthorized software besides malware). This software can pose multiple problems. For example, it can introduce unmanaged vulnerable software into the environment, which can then be used by attackers to exploit hosts and further compromise them. There can also be legal issues with the installation of unauthorized software, such as violations of licensing agreements.

Application whitelisting is most readily used to stop threats on managed hosts where users are not able to install or run applications without authorization. An example is a kiosk workstation where users are limited to running a web browser; installation and execution of all applications other than the selected web browser and authorized application-based security controls (such as antivirus software) would be prohibited. Another example is a laptop that has all authorized applications preinstalled for the user, and the user does not have the administrative privileges necessary to install additional applications or disable the application whitelisting software. Application whitelisting may also be beneficial on servers, particularly if there is concern about malware spreading to these servers from other hosts (e.g., administrator laptops).

2.2 Types of Application Whitelisting

This section discusses the types of application whitelisting. This includes the application file and folder attributes that can be analyzed; the types of application resources handled, such as executables, libraries, and scripts; and techniques for whitelist generation.

2.2.1 File and Folder Attributes

Application whitelisting can be based on a variety of application file and folder attributes, including the following:[2]

- **File path.** This is the most general attribute: to permit all applications contained within a particular path (directory/folder). Used by itself, this is a very weak attribute, because it allows any malicious files placed within the directory to be executed. However, if the path is protected by strict access controls that only allow authorized administrators to add or modify files, this becomes a stronger attribute. Paths can be beneficial by not requiring each file within the path to be listed separately, which reduces the need to update the whitelist for every new application and patch.
- **Filename.** This attribute, for the name of an application file, is too general to be used on its own. If a file were to become infected or be replaced, its name would be unchanged so the file would still be executed under the whitelist. Also, an attacker could simply place a malicious file onto a host and use the same name as a common benign file. Because of these weaknesses, this attribute should not be used on its own; rather, it should be paired with other attributes. For example, it would be stronger to combine path and filename attributes with strict access controls or to combine a filename attribute with a digital signature attribute (described below).
- **File size.** This attribute is typically used only in combination with other attributes, such as filename. Monitoring the file size assumes that a malicious version of an application would have a different file size than the original; however, attackers can craft malicious files to have the same length as their benign counterparts. Other attributes, such as digital signature and cryptographic hash, provide

[2] This list of attributes is not intended to be all-inclusive. Also, it is expected that new forms of attributes may arise as technologies advance. Another set of attributes is the *software identification tags (SWID tags)* which define unique information about an installed software application, including its name, edition, version, whether it's part of a bundle and more (http://tagvault.org/swid-tags/).

substantially better unique identification of files than file size does, and should be used instead of file size whenever feasible.

- **Digital signature or publisher.** A digital signature can provide a reliable, unique value for an application file, assuming that the signature is verified to ensure that it is legitimate. The only exception is if a publisher's signing key is compromised. Unfortunately, while application files are increasingly being digitally signed by their publishers, many are not yet signed, so using only publisher-provided digital signatures as attributes is generally not feasible. Some application whitelists can be based on publisher, not just individual digital signatures; this is based on the assumption that all applications from trusted publishers can themselves be trusted.[3] This assumption may be faulty if the software vendor has multiple applications and the organization wants to restrict which of those applications can be executed. Also, relying on publisher only would allow older software versions with known vulnerabilities to be executed. However, the benefit of basing a whitelist on publisher is that the whitelist only needs updates when there is a new publisher (i.e., software vendor) or when a publisher updates its signature key.[4]

- **Cryptographic hash.** A cryptographic hash provides a reliable, unique value for an application file, so long as the cryptography being used is strong and the hash is already known to be associated with a good file. Cryptographic hashes are accurate no matter where the file is placed, what it is named, or how it is signed. However, cryptographic hashes are not helpful when a file is updated, such as an application being patched; the patched version will have a different hash value. In these cases the patch should be identified as legitimate through its digital signature, then its cryptographic hash should be added to the whitelist. Note that if the whitelist is not continuously updated with new hashes for new and updated applications, there is a significant risk of software not functioning correctly.

As the discussions above indicate, choosing attributes is largely a matter of achieving the right balance of security, maintainability, and usability. Simpler attributes such as file path, filename, and file size should not be used by themselves unless there are strict access controls in place to tightly restrict file activity, and even then there are often significant benefits to pairing them with other attributes. A combination of digital signature/publisher and cryptographic hash techniques generally provides the most accurate and comprehensive application whitelisting capability, but usability and maintainability requirements can put significant burdens on the organization.

2.2.2 Application Resources

Application whitelisting is most often associated with monitoring executables. However, most application whitelisting technologies also have the ability to monitor at least some other types of application-related files, such as libraries, scripts, macros, browsers plug-ins or add-ons or extensions, and configuration files, plus application-related registry entries on Windows hosts. The granularity of this monitoring varies significantly among application whitelisting technologies; for example, some can only permit or block whole classes of scripts (e.g., JavaScript)[5], while others can permit or block individual scripts within a class of scripts.

---

[3] For its internal applications, an organization can issue its own internal signing key to anchor its root of trust, instead of depending on a signing key from an external publisher.

[4] An alternative approach is to employ cross-signing, where both the software vendor and the organization sign each application, thus indicating that it is both authentic and approved by the organization.

[5] Generally this means that the application is blocking the executable for the scripting language, instead of blocking the scripts themselves.

2.2.3 Whitelist Generation and Maintenance

There are two primary methods of generating an application whitelist for a host. One is to use vendor-provided information on the characteristics of known good applications, supplemented with organization-generated information on the characteristics of organization-specific applications (i.e., in-house custom applications). The other method of generating an application whitelist is to scan the files on a clean host to build a good known baseline.[6]

Both of these methods are effective on their own, except when applications are updated (e.g., patched) or new applications are installed. If the vendor is providing the whitelist information, the vendor will have to acquire the patch or new application, record its files' characteristics, and send the corresponding information to customers. If the organization is building its own whitelist information, it will have to: acquire each patch or new application, record its files' characteristics, and update its whitelists with the new information; or, redo its known good baseline to serve as the new reference baseline. Any of these methods may cause problematic delays for organizations that apply patches quickly, especially automatically; patched software may be seen as unknown software and prohibited from running. Certain attributes, such as file path and publisher, generally do not change with each patch and so whitelists utilizing those attributes do not need to be updated as often and should cause fewer of these delays.

To avoid these problems with updates, most application whitelisting technologies offer maintenance options. For example, many technologies allow the administrator to select certain services (e.g., patch management software) to be trusted updaters. This means that any files that they add to or modify on a host are automatically added to the whitelist. Similar options are available for designating trusted publishers (i.e., software vendors), users (such as system administrators), sources (such as trusted network paths), and other trusted entities that may update whitelists.

Another option available with some application whitelisting technologies is the use of reputation services. These services determine if a service, publisher, or other external entity is generally associated with benign or malicious content. This allows application whitelisting software to make decisions about how to handle new or modified files based on the reputation of the associated service, publisher, etc., instead of simply adding them to a graylist for subsequent manual processing.

2.3 Application Whitelisting Modes

Most application whitelisting technologies offer two operational runtime modes:

- **Audit mode**, which allows items, including those not on the whitelist to be executed and logs their execution.
- **Enforcement mode**, which blocks execution of unapproved items. Enforcement mode automatically permits execution of whitelisted items and blocks execution of blacklisted items. There are different forms of enforcement mode, which are differentiated by how they handle items that are not whitelisted or blacklisted. These forms include the following:
  - **Whitelist enforcement**, which permits only whitelist items to be executed and blocks execution of all others.
  - **User prompting**, which asks the user (or, in some cases, the administrator) to accept or reject each attempt to execute a file that is not whitelisted or blacklisted.

[6] NIST hosts the National Software Reference Library (NSRL), which contains metadata for application files for forensic investigation purposes. See http://www.nsrl.nist.gov/ for additional information.

- **Blacklist enforcement**, which allows everything to be executed that is not blacklisted.

An application whitelisting technology run in audit mode is strictly informative; it can log the execution of malware and other unauthorized executables, but it cannot do anything to stop them. Audit mode is primarily intended for use when first deploying an application whitelisting technology, to help an organization evaluate and fine-tune the technology before switching it to enforcement mode.

Many application whitelisting technologies have granular options for setting modes. Some features could be configured to run in enforcement mode while other features run in audit mode. For example, Windows registry changes might be permitted (audit mode) while operating system file changes would be prohibited (enforcement mode). Some products also support multiple enforcement modes and allow granular setting of those for different types of monitored entities.

2.4 Uses of Application Whitelisting Technologies

As stated in the Section 2 introduction, the primary purpose of application whitelisting technologies is to provide application access control: to stop the execution of unauthorized software. However, most application whitelisting technologies can be used for other purposes as well, including the following:

- **Software inventory.** Application whitelisting technologies can keep an inventory of which applications and application versions are installed on each host. This allows an organization to identify unauthorized applications—unlicensed applications, prohibited applications, etc.—as well as to identify "wrong" versions of software (both too old and too new). This software inventory capability is also useful for forensic investigations, such as finding modified applications, unauthorized applications, malware, unknown applications, etc. on a given host.
- **File integrity monitoring.** Most application whitelisting technologies can perform frequent or continuous monitoring of attempted changes to application files. Some technologies can prevent files from being changed, while other technologies cannot prevent changes but can immediately report when changes occur.
- **Incident response.** An organization responding to an incident on a host could capture the characteristics of the malicious files on that host (e.g., generate cryptographic file hashes) and use application whitelisting technologies to check other hosts for the same files, indicating that they have been compromised as well.

Some application whitelisting technologies may have additional capabilities, including the following:

- Access control for portable storage devices, such as restricting file reads, writes, and executes for all files on removable media; only permitting the use of encrypted devices; and only permitting use of drives with particular serial numbers.
- Memory protection, primarily involving stopping certain attacks (e.g., buffer overflows) that directly affect files in memory, not files in storage. Most application whitelisting technologies only focus on the files in storage, but do not ensure that the files in memory are not altered or exploited.
- Software reputation services, such as reviewing what other software a particular application is often bundled with, and determining if an application is known to pose a substantial security risk.
- Antivirus scanner integration; an example is running graylisted files through an online scanner with many antivirus scanning engines to attempt to determine if the files are known to be malicious.

2.5 Operational Environment Differences

As discussed in NIST SP 800-70, *National Checklist Program for IT Products—Guidelines for Checklist Users and Developers*,[7] there are significant differences among operational environments. These differences are important in terms of selecting and deploying application whitelisting technologies. The major categories of operational environments are as follows:

- **Standalone.** Also referred to as **Small Office/Home Office (SOHO)**, a Standalone environment refers to a small, informal computer installation that is used for home or business purposes. For technical and business (economic) reasons, Standalone environment hosts are generally not managed remotely. Standalone environments are typically the least secured.
- **Managed**. The Managed environment, also called an **Enterprise** environment, typically contains large organizational systems with defined suites of hardware and software configurations, usually consisting of centrally managed IT products (e.g., workstations and servers). The managed nature of these environments gives administrators centralized control over various settings on IT products. Because of the supported and largely homogeneous nature of the Managed environment, it is typically easier to use more functionally restrictive settings in Managed environments than in Standalone environments.
- **Specialized Security-Limited Functionality (Custom)**. A Custom environment contains systems in which the functionality and degree of security do not fit the Standalone or Managed environments. Specialized Security-Limited Functionality (SSLF) is a Custom environment that is highly restrictive and secure; it is usually reserved for hosts that have the highest threats and associated impacts. Because hosts in an SSLF environment are at high risk of attack or data exposure, security takes high precedence over functionality.

The first step in evaluating the possibility of deploying an application whitelisting solution should be an analysis of the environment or environments in which the hosts will be running. Generally it is not feasible to implement whitelisting on Standalone environment hosts because of the lack of centralized management. Application whitelisting solutions are generally strongly recommended for hosts in SSLF environments because of the high risks that they face. Suitability for Managed environments depends on how tightly the hosts are managed and the extent of the risks that they face; organizations considering application whitelisting deployment in a Managed environment should perform a risk assessment to determine whether the security benefits provided by application whitelisting outweigh its possible negative impact on operations. Organizations should also be mindful that they will need dedicated staff managing and maintaining the application whitelisting solution, similar to handling an enterprise antivirus or intrusion detection solution.

Once it has been determined that application whitelisting technologies are merited for a particular environment, the next step is to consider which technologies might be feasible. Organizations should consider application whitelisting technologies already built into the operating system, particularly for centrally managed hosts, because of the relative ease and minimal additional cost in managing these solutions. If built-in application whitelisting capabilities are not available or are determined to be unsuitable, then the alternative is to examine third-party solutions with robust centralized management capabilities. An organization that can dedicate the necessary trained staff to solution maintenance and has built-in application whitelisting technology should generally implement application whitelisting at least in a monitoring mode.

It is highly recommended to test any prospective application whitelisting technology in a monitoring

[7] http://csrc.nist.gov/publications/nistpubs/800-70-rev2/SP800-70-rev2.pdf

mode to see how it would behave before solution deployment. This testing should include a thorough evaluation of how the solution reacts to changes in software, such as installing an update. An application whitelisting technology might be considered unsuitable if, for instance, it had to be disabled in order to install security updates for the operating system or particular applications.

2.6 Additional Considerations

This section describes additional considerations that organizations should examine when evaluating the likely effectiveness of potential application whitelisting technology solutions.

Effectiveness Consideration	Further Explanation
How easily can a solution be bypassed?	If a solution can be bypassed easily, some users will choose to do so in order to run unauthorized software, and malware may take advantage of the configuration weakness to execute on the host.
How complex is a solution (hash-based versus signature-based, etc.)?	Generally, more complex solutions will be harder for an attacker to circumvent. A relatively simple solution lacks the features necessary to minimize false positives and false negatives. However, more complex solutions may have higher administrative and maintenance overhead.
What are the relative costs of a solution?	Solutions built into the operating system might be significantly less expensive to operate than third-party solutions. The cost of third-party solutions (including licensing and support) may vary widely.
What impact does the solution have on standard performance?	Using application whitelisting technologies generally should not be noticeable to users in terms of significantly slowing host performance.
What impact does the solution have on business/mission?	If the product does not minimize false positives, users may frequently being prevented from running authorized software. If the product does not minimize false negatives, malware infections are more likely to occur. Both of these circumstances could seriously impact the organization's mission, depending on the value of the relevant hosts.
How usable is the solution for both users and administrators?	A more usable solution will not only minimize false positives, to minimize user disruption, but it will also provide pertinent information to users and administrators when software is blocked from installation or execution.
What are the long-term maintenance demands for running the solution?	As new applications are added to the environment and existing applications are updated, there may be technical difficulties in keeping whitelists updated in a timely manner, and significant costs associated with maintenance. Certain types of whitelisting require more frequent whitelist changes than others. However, the amount of maintenance needed must be balanced with the effectiveness of the solution; a higher-maintenance solution that prevents more incidents may actually be less expensive overall including the cost to remediate incidents than a lower-maintenance solution that has limited effectiveness in stopping threats.

3. Application Whitelisting Planning and Implementation

This section discusses considerations for planning and implementing application whitelisting technologies for end user devices. As with any new technology deployment, application whitelisting technology planning and implementation should be addressed in a phased approach. A successful deployment can be achieved by following a clear, step-by-step planning and implementation process. The use of a phased approach for deployment can minimize unforeseen issues and identify potential pitfalls early in the process. This model also allows for incorporating advances in new technology and adapting the technology to the ever-changing enterprise. The following is an example of planning and implementation phases:

1. **Initiate the Solution.** The first phase involves identifying current and future needs for application whitelisting; specifying requirements for performance, functionality, and security; and developing necessary policies.

2. **Design the Solution.** The second phase involves all facets of designing the application whitelisting solution. Examples include architectural considerations, whitelist management, cryptography policy, and security aspects of the solution itself.

3. **Implement and Test a Prototype.** The next phase involves implementing and testing a prototype of the designed solution in a lab or test environment. The primary goals of the testing are to evaluate the functionality, management, performance, and security of the solution.

4. **Deploy the Solution.** Once the testing is completed and all issues are resolved, the next phase includes the gradual deployment of the application whitelisting technology throughout the enterprise.

5. **Manage the Solution.** After the solution has been deployed, it is managed throughout its lifecycle. Management includes solution maintenance and support for operational issues. The lifecycle process is repeated when enhancements or significant changes need to be incorporated into the solution.

This document does not describe the planning and implementation process in depth because the same basic steps are performed for any security technology. This section only highlights those considerations that are of particular interest for application whitelisting technologies. These considerations are not intended to be comprehensive, nor is there any implication that particular security elements not listed here are unimportant or unnecessary. In addition to following the security recommendations presented in this publication, organizations implementing application whitelisting technologies should also follow the recommendations from NIST Special Publication (SP) 800-53, *Recommended Security Controls for Federal Information Systems and Organizations*, which defines minimum recommended management, operational, and technical controls for information systems based on impact categories.

3.1 Initiation

The purpose of this phase is to identify the current and future needs for application whitelisting and to determine how those needs can best be met. Requirements specific to application whitelisting that should be considered include the following:

- **External Requirements.** The organization may be subject to oversight or review by another organization that requires application whitelisting.

- **System and Network Environments.** It is important to understand the characteristics of the organization's system and network environments so that application whitelisting solutions can be selected that will be compatible with them and able to provide the necessary functionality. Aspects to consider include the following:
  - The characteristics of the devices that need application whitelisting, especially the OSs and applications
  - The technical attributes of the interfaces of other systems with which the application whitelisting solution might be integrated, such as centralized logging servers and security information and event management (SIEM) software

The outcome of the organization's analysis of requirements should be a determination of which types of applications or application components (executables, libraries, registry entries, configuration files, etc.) need to be monitored, which types of threats the application whitelisting should protect against, and which types of application whitelisting should be used to balance security, usability, and maintainability. For example, the organization may decide to block execution of all unauthorized application components on higher-risk client systems, while monitoring (but not blocking) execution of unauthorized application components on lower-risk client systems. These decisions should be captured in policy.

Another outcome of the analysis is the documentation of the requirements for the application whitelisting technologies themselves, including security capabilities (e.g., authentication, cryptography, key management), performance requirements, management requirements (including reliability, interoperability, and scalability), the security of the technology itself, usability, and maintenance requirements (such as applying updates).

In many cases, a single application whitelisting product cannot meet all of the organization's identified needs. For example, the organization may need to monitor applications on devices running several different OSs, yet no appropriate product can work on all those platforms. Also, some operating systems may have application whitelisting technologies built-in. Organizations can solve this problem in several ways, such as acquiring multiple products or replacing older devices. Organizations should ensure that effective solutions are identified for all the types of end user devices that need their applications monitored, if possible, and that a waiver and risk management process is created for unusual cases that cannot be addressed by the identified solutions.

Examples of challenging platforms for application whitelisting include mobile devices[8] and industrial control systems (ICS)[9]. One of the main benefits of using mobile devices is being able to acquire a wide variety of applications easily, quickly, and cheaply (often free). Unfortunately, this philosophy makes it infeasible in many cases to implement whitelisting for mobile devices. If mobile devices are tightly managed, much like some desktops or laptops, and only allowed to acquire approved apps from an enterprise-sponsored app store, then whitelisting may be practical, but for user-controlled unmanaged mobile devices, whitelisting may not be an option as of this writing.

An ICS is a challenging platform for whitelisting in part because, unlike most other computing devices, ICSes strongly favor availability over confidentiality. It is of utmost importance that ICSes continue to function properly no matter what is happening to them, including cyber attacks. Because application whitelisting can inadvertently prevent benign executables from being run, its use for ICSes must be

[8] For more information on mobile device security, see NIST SP 800-124, *Guidelines for Managing the Security of Mobile Devices in the Enterprise* (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf).

[9] More information on ICS is available from NIST SP 800-82, *Guide to Industrial Control Systems (ICS) Security* (http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf).

carefully analyzed and tested for feasibility. Another problem with ICSes is that they often use atypical platforms, which may not be supported by any acceptable application whitelisting solutions. However, since ICSes are used for specific functionality and run only certain ICS software, in some cases they are actually easier to whitelist than more dynamic, heterogeneous environments.

3.2 Design

Once the needs have been identified and the appropriate application whitelisting technologies have been chosen, the next phase is to design a solution that meets those needs. If these design decisions are incorrect, then the application whitelisting implementation will be more susceptible to compromise and failure. Major aspects of solution design that are particularly important for application whitelisting are as follows:

- **Cryptography.** Cryptography is used in at least three ways for application whitelisting technologies: to generate and verify cryptographic hashes for files and other application components, to validate digital signatures for files, and to protect the confidentiality and integrity of communications between individual hosts and centralized management (for example, encrypting lists of installed applications and application versions). For all of these functions, Federal agencies must use Federal Information Processing Standard (FIPS) approved algorithms contained in validated cryptographic modules. Organizations should consider how easily the solution can be updated when stronger algorithms and key sizes become available in the future.
- **Solution architecture.** The architecture of the application whitelisting technology refers to the selection of devices and software to provide application whitelisting services and the placement of centralized elements within the existing network infrastructure, such as management servers. Most application whitelisting technologies can only operate as a centrally managed solution; there may be copies of whitelists on individual hosts, but enterprise management is centralized. Each end user device must have software that provides application whitelisting enforcement or auditing. Designing the architecture includes component placement, redundancy, reliability, and interoperability.
- **Whitelist management.** As discussed in Section 2.1.3, whitelist management can involve the establishment of trusted publishers, users, updaters, etc. Organizations should choose these trusted entities carefully because a compromise in a trusted entity could lead to the compromise of the application whitelisting technology, and consequently the hosts it protects. However, failure to include necessary entities as trusted will likely lead to operational problems, such as when patch installation is not automatically trusted by the application whitelisting technology.

3.3 Prototype Testing

After the solution has been designed, the next step is to implement and test a prototype of the design. Ideally, implementation and testing should first be performed on lab or test devices. Only implementations in final testing should be conducted on production devices. Aspects of the solution to evaluate include the following:

- **Application control functionality.** Basic functionality will need to be tested during the prototype testing. Examples are allowing whitelisted applications to be run, blocking blacklisted applications from running, and detecting modifications to whitelisted applications. These functions should be verified by: installing patches and other updates; manually modifying executables; and making other changes to applications to confirm that the application control policies can be properly enforced and cannot be easily circumvented.

- **Management.** Administrators should be able to configure and manage all components of the solution effectively and securely. It is particularly important to evaluate the ease of deployment and configuration, including how easily the solution can be managed as the solution is scaled to larger deployments. Management concerns should include the effects of patching/upgrading the application whitelisting software, changing software settings (e.g., changing cryptographic algorithms or key sizes), and managing cryptographic keys. Another important management concern that needs special attention is whitelist generation and maintenance, such as how the whitelists accommodate software patching.
- **Logging/alerting.** The logging, alerting, and data management functions should work properly in accordance with the organization's policies and strategies.
- **Performance.** The solution should be able to provide adequate performance during normal and peak usage. Testing should incorporate a variety of devices, OSs, and applications.
- **Security of the implementation.** The application whitelisting technology itself may contain vulnerabilities and weaknesses that attackers could exploit. Organizations with high security needs may want to perform extensive vulnerability assessments against the application whitelisting components.

Before installing application whitelisting software on a host, organizations should scan the host for malware and either remove any malware that is detected or rebuild the host. The scan will ensure that malware files are not included in the whitelist generation process. Organizations should also ensure that the host's OS is secured properly, including that it is fully patched and that other necessary security controls are installed and configured properly. If the OS is not secured properly, the host is more likely to be compromised, which could weaken the protection provided by the application whitelisting technology.

3.4 Deployment

Once testing is complete and any issues have been resolved, the next phase of the planning and implementation model involves deploying the solution. When the components are being deployed into production, organizations should initially use application whitelisting on a small number of hosts. Deploying it to many hosts at once might overwhelm the management servers or identify other bottlenecks through loss of availability. Many of the problems that occur are likely to occur on multiple hosts, so it is helpful to identify such problems either during the testing process or when deploying the first hosts, so that those problems can be addressed before widespread deployment. A phased deployment provides administrators an opportunity to evaluate the impact of the solution and resolve issues prior to enterprise-wide deployment. It also provides time for the IT staff (e.g., system administrators, help desk) and users to be trained and to become accustomed to the operational lifecycle of the implementation.

Most of the issues that can occur during deployment are the same types of issues that occur during any large IT deployment. In addition to potential problems described earlier in this publication, another typical issue is end users discovering and disabling the application whitelisting software. Many products run in a stealth mode so that users cannot readily tell that they are running.

3.5 Management

The last phase of the planning and implementation model is the longest lasting. Managing the solution involves operating the deployed solution and maintaining the application whitelisting architecture, policies, software, and other solution components. Examples of typical actions include:

- Updating the whitelist for new or updated applications
- Testing and applying patches to the application whitelisting software
- Deploying application whitelisting to additional platforms
- Performing key management duties
- Adapting policies as requirements change
- Monitoring the components for operational and security issues
- Periodically performing testing to ensure that application whitelisting is functioning properly
- Performing regular vulnerability assessments

Organizations should pay particular attention to the ongoing whitelist updates. Although many, if not most, whitelist updates can be automated, administrators should be prepared to make manual updates quickly when needed, such as to identify emerging threats and to correct false positives or negatives. Organizations should also monitor any graylists and transfer their entries to whitelists or blacklists, as appropriate.

Appendix A—Security and Compliance Mapping

This appendix provides a mapping to selected standards and guidelines that support using application whitelisting technologies.

NIST Special Publication 800-53 Revision 4, *Security and Privacy Controls for Federal Information Systems and Organizations*[10]

- Control CM-7 (Least Functionality), control enhancement 5 (Authorized Software/Whitelisting): "The organization:
  - (a) Identifies [Assignment: organization-defined software programs authorized to execute on the information system];
  - (b) Employs a deny-all, permit-by-exception policy to allow the execution of authorized software programs on the information system; and
  - (c) Reviews and updates the list of authorized software programs [Assignment: organization-defined frequency]."

***Framework for Improving Critical Infrastructure Cybersecurity, Version 1.0*[11]**

- Subcategory PR.IP-1:[12] "A baseline configuration of information technology/industrial control systems is created and maintained." This refers to determining which applications are authorized to be run on each system.
- Subcategory PR.PT-3:[13] "Access to systems and assets is controlled, incorporating the principle of least functionality." This refers to the enforcement of the whitelist established through subcategory PR.IP-1.

***Critical Controls for Effective Cyber Defense, Version 5*[14]**

- Critical Control 2: Inventory of Authorized and Unauthorized Software: "Actively manage (inventory, track, and correct) all software on the network so that only authorized software is installed and can execute, and that unauthorized and unmanaged software is found and prevented from installation or execution."

---

10 http://dx.doi.org/10.6028/NIST.SP.800-53r4

11 http://www.nist.gov/cyberframework/index.cfm

12 PR.IP stands for Protect: Information Protection Processes and Procedures. This is defined as follows: "Security policies (that address purpose, scope, roles, responsibilities, management commitment, and coordination among organizational entities), processes, and procedures are maintained and used to manage protection of information systems and assets."

13 PR.PT stands for Protect: Protective Technology (PT). This is defined as follows: "Technical security solutions are managed to ensure the security and resilience of systems and assets, consistent with related policies, procedures, and agreements."

14 http://www.counciloncybersecurity.org/critical-controls/reports/

Appendix B—Building Block

This appendix contains a detailed discussion of a particular opportunity that is relevant across a variety of industry sectors. The discussion essentially constitutes a "building block" for addressing the challenge.

B.1 Description

Malware has evolved over time to present increasingly customized, targeted threats to individual users. These threats are much harder to detect than previous generations of malware when using traditional antimalware controls, such as antivirus software and intrusion detection systems (IDS), because these controls are largely signature-based, recognizing only known characteristics of known threats. Customized, targeted malware is specifically intended to be unique to each instance, and therefore challenging to identify based on prior analysis of previous instances as being malicious in nature.

An alternative to traditional antimalware controls that is increasing in popularity is application whitelisting. Application whitelisting allows only those applications that are known to be benign to be executed on a host. Unknown applications, including all malware, are prohibited from execution, thus preventing infections and compromises. The primary drawback with application whitelisting is its maintenance and usability implications—for example, an update to an application may cause application whitelisting software to suddenly start treating the application as unknown and therefore unauthorized. So the goal is to have an application whitelisting solution that prevents malware-based compromises while minimizing the maintenance burden and ensuring that the resulting host environment is highly usable.

Example Scenario 1 – Operating System Updates

An organization issues centrally managed laptops to many of its employees. These laptops have a standard configuration, and users are not authorized to make administrative changes to them. The operating system is patched as part of an enterprise patch management solution. To prevent malware-based compromises, each laptop runs third-party antivirus software and uses application whitelisting technology built into the operating system. As patches are deployed to operating systems, the application whitelisting solution is updated either simultaneously or previously so that the new "version" of the operating system is recognized by the application whitelisting technology. After patch deployment, the user notices no negative impact caused by the application whitelisting.

Example Scenario 2 – Application Updates

An organization issues centrally managed laptops to many of its employees. These laptops have a standard configuration, and users are not authorized to install additional applications or make other administrative changes to them. The organization's standard client applications are configured to be updated as part of an enterprise patch management solution. Non-standard client applications (those additional applications placed onto the host by an authorized system administrator) typically patch themselves (e.g., are configured to automatically download and install updates as they become available). As patches are applied to standard and non-standard client applications, the application whitelisting solution built into these laptops adjusts itself as needed to continue to recognize the current version of approved applications as being valid and not permitting any other applications or old vulnerable application versions to run.

B.2 Business Drivers

- Enable adoption of new and updated applications

- Prevent unlicensed, vulnerable, and otherwise unauthorized applications from being executed within the organization, thus preventing potential compromises of organization data and resources
- Compensate for the reduced effectiveness of antivirus software and IDS technologies

B.3 Approach

This building block is intended to demonstrate security capabilities that can provide greater assurance that a client device, such as a laptop or desktop, can be free of malware. As described throughout this publication, supplementing antivirus software and other antimalware controls with application whitelisting software is proposed as the fundamental approach to solving this problem. The favored method of application whitelisting is capabilities that are built into the local operating system.

The most important security control in terms of interactions with the application whitelisting is security updates, particularly patch management software. If the application whitelisting configuration is not harmonious with the patch management software, it is inevitable that there will be significant usability problems when newly updated applications are not recognized as authorized and are not allowed to execute.

Another important element of the application whitelisting software approach is that the whitelisting software needs to be configured to reduce the likelihood of attackers circumventing it. The solution that most strongly protects against circumvention may also require the most maintenance to keep up with patches and other updates, so maintenance must be taken into account in any approach.

B.4 Desired Security Capabilities

- Only authorized applications (including operating system components) are allowed to execute. All other applications are prohibited from executing.
- Older versions of an application with known vulnerabilities can be prevented from executing to prevent potential compromises.
- Patch management should not cause any delay in application execution other than what is necessary for the patching process itself (e.g., replacing executables, rebooting the host). There should be no extended delays because an application whitelisting solution does not recognize a new version of an authorized application.

B.5 Business Value

- Prevents execution of malware, which could compromise sensitive organization data.

B.6 Relevant Standards

- NIST SP 800-83 Rev.1, Guide to Malware Incident Prevention and Handling for Desktops and Laptops
  http://csrc.nist.gov/publications/PubsSPs.html#800-83
- National Security Agency Top IA Mitigation Strategies - Application Whitelisting
  http://www.nsa.gov/ia/_files/factsheets/I43V_Slick_Sheets/SlickSheet_ApplicationWhitelisting_Standard.pdf

Appendix C—Applying Application Whitelisting to Mobile Platforms

This appendix discusses considerations involved in applying application whitelisting to mobile platforms (e.g., smartphones, tablets). Typical standalone application whitelisting technologies are generally not available for mobile devices as of this writing. Instead, application whitelisting is achieved through one of two methods: mobile device management (MDM)/mobile application management (MAM) or an enterprise app store.

MDM/MAM

MDM[15] and MAM technologies are suites of security controls for protecting mobile devices from compromises. MDM and MAM technologies often have application whitelisting capabilities built in. Because MDM and MAM technologies are typically centrally managed, they offer a relatively easy way to deploy whitelisting capabilities to mobile devices. However, the disadvantage of relying on application whitelisting in this environment is that mobile applications are constantly changing and new applications are released all the time; it may be prohibitively difficult to maintain application whitelisting solutions with that much flux to be addressed.

Enterprise App Store

An alternative to a client-based application whitelisting technology is an enterprise app store[16]. Many organizations, especially those with MDM deployed to their mobile devices, control which app stores their users may download and install apps from. This effectively provides a form of application whitelisting, because only those applications that have been approved by the organization for inclusion in the app store may be accessed by the organization's users. There is some maintenance overhead associated with relying on an app store for whitelisting, but it is centralized (approving an app once and posting it to the app store) instead of distributed (configuring thousands of managed mobile devices to recognize the latest apps and app updates).

[15] NIST SP 800-124 Guidelines for Managing the Security of Mobile Devices in the Enterprise - http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf

[16] Draft NIST SP 800-163 Technical Considerations for Vetting 3rd Party Mobile Applications – http://csrc.nist.gov/publications/drafts/800-163/sp800_163_draft.pdf

Appendix D—Acronyms and Abbreviations

Selected acronyms and abbreviations used in the guide are defined below.

FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
ICS	Industrial Control Systems
IT	Information Technology
ITL	Information Technology Laboratory
MAM	Mobile Application Management
MDM	Mobile Device Management
NIST	National Institute of Standards and Technology
NSRL	National Software Reference Library
OMB	Office of Management and Budget
OS	Operating System
SIEM	Security Information and Event Management
SP	Special Publication
SSLF	Specialized Security-Limited Functionality

Appendix E—Reference

The references for the publication are listed below.

- National Security Agency, "Top IA Mitigation Strategies - Application Whitelisting" http://www.nsa.gov/ia/_files/factsheets/I43V_Slick_Sheets/SlickSheet_ApplicationWhitelisting_Standard.pdf
- National Security Agency, "Application Whitelisting Using Software Restriction Policies" http://www.nsa.gov/ia/_files/os/win2k/application_whitelisting_using_srp.pdf