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

NIST Interagency Report 7343
(Draft)

The Security Content Automation Program (SCAP): Automating Compliance Checking, Vulnerability Management, and Security Measurement

Stephen D. Quinn
Peter Mell
Karen Kent

NIST Interagency Report 7343 (Draft)

The Security Content Automation Program (SCAP): Automating Compliance Checking, Vulnerability Management, and Security Measurement (Draft)

Stephen D. Quinn
Peter Mell
Karen Kent

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

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

October 2006

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 analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 7343 (Draft)
44 pages (October 2006)

Abstract

A security checklist is a document that contains instructions for securely configuring an information technology (IT) product for an operational environment or verifying that an IT product has already been securely configured. Checklists can take many forms, including files that can automatically set or verify security configurations; having such automated methods has become increasingly important for several reasons, including the complexity of achieving compliance with various laws, regulations, and guidelines, and the increasing rates of vulnerabilities in systems and threats against those vulnerabilities. Automation is also needed to ensure that systems are secured consistently and their security verified effectively.

In response to these needs, the Security Content Automation Program (SCAP) seeks to encourage the development of automated checklists, particularly those that are compliant or compatible with the Extensible Configuration Checklist Description Format (XCCDF) and/or the Open Vulnerability and Assessment Language (OVAL). These are widely used for automated checklists—XCCDF primarily for mapping policies and other sets of requirements to high-level technical checks, and OVAL primarily for mapping high-level technical checks to the low-level details of executing those checks. For example, XCCDF could map a requirement for authentication management in NIST Special Publication (SP) 800-53 to a specified need to check that the system's minimum password length is at least 8 characters. OVAL could then define how that check should be performed on a particular type of system, such as a Windows computer or a UNIX computer.

This publication provides an overview of the Security Content Automation Program, and then focuses on explaining how XCCDF and OVAL files can be used as part of security content automation. The publication also examines how security content automation can be beneficial in achieving compliance with the Federal Information Security Management Act (FISMA), and how some of the same checklist components used to support FISMA compliance efforts can also be used for other compliance needs, such as Department of Defense (DOD) 8500.2/8510 compliance.

Authority

The National Institute of Standards and Technology (NIST) developed this document in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets; but such standards and guidelines shall not apply to national security 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 A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright, though attribution is desired.

Nothing in this document should be taken to contradict 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.

Purpose and Scope

The Cyber Security Research and Development Act of 2002 tasks NIST to develop, and revise as necessary, a checklist setting forth settings and option selections that minimize the security risks associated with each computer hardware or software system that is, or is likely to become widely used within the Federal government. Such checklists, when combined with well-developed guidance, leveraged with high-quality security expertise, vendor product knowledge, operational experience, and accompanied with tools, can markedly reduce the vulnerability exposure of an organization.

This publication is intended to educate readers on the first planned phase of the Security Content Automation Program (SCAP). The program seeks to encourage the development of checklists that can be used with a variety of tools to automate the application or verification of security-related configuration settings for operating systems and application. SCAP specifically focuses on the creation of checklists that support agency compliance with FISMA and map to the minimum security controls for information systems described in NIST Special Publication (SP) 800-53, *Recommended Security Controls for Federal Information Systems*, and the information system categories described in Federal Information Processing Standards (FIPS) 199, *Standards for Security Categorization of Federal Information and Information Systems.*[1]

Audience

The primary audience for this publication is computer security staff, system administrators, IT product vendors, security checklist developers, and others who are responsible for performing duties involving the use of security checklists to apply security settings or verify existing settings. Those responsible for developing policy and security guidance should also find this document helpful.

[1] The text in the Purpose and Scope section is based on text from NIST Interagency Report (IR) 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF)*.

Acknowledgements

The authors, Stephen D. Quinn, Peter Mell, and Karen Kent of the National Institute of Standards and Technology (NIST), wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Tim Grance, Ron Ross, and Murugiah Souppaya from NIST, as well as representatives from MITRE, Secure Elements, and Threat Guard, for their keen and insightful assistance throughout the development of the document. Additional acknowledgements will be added to the final version of the publication.

Trademark Information

Table of Contents

List of Appendices

List of Figures

List of Tables

1. Introduction to Security Checklists

A *security checklist* (sometimes referred to as a lockdown guide, hardening guide, security guide, Security Technical Implementation Guide [STIG], security configuration guide, or benchmark)[2] is essentially a document that contains instructions or procedures for securely configuring an information technology (IT) product for an operational environment or verifying that an IT product has already been securely configured. A checklist might include any of the following:

- Configuration files that automatically set or verify various security settings (e.g., executables, security templates that modify settings, scripts)
- Documentation (e.g., text file) that guides the checklist user to manually configure an IT product
- Documents that explain the recommended methods to securely install and configure a device
- Policy documents that set forth guidelines for such things as auditing, authentication mechanisms (e.g., passwords), and perimeter security.

Not all instructions in a security checklist are necessarily for security settings. Checklists can also include administrative practices for an IT product that go hand-in-hand with improvements to the product's security. Often, successful attacks on systems are the direct result of poor administrative practices such as not changing default passwords or failure to apply old patches.

The following are some examples of the types of devices and software for which checklists are intended:

- General-purpose operating systems
- Common desktop applications such as e-mail clients, Web browsers, word processors, personal firewalls, and antivirus software
- Infrastructure devices such as routers, firewalls, virtual private network (VPN) gateways, intrusion detection systems (IDS), wireless access points, and telecom systems
- Application servers such as Domain Name System (DNS) servers, Dynamic Host Configuration Protocol (DHCP) servers, Web servers, Simple Mail Transfer Protocol (SMTP) servers, File Transfer Protocol (FTP) servers, and database servers
- Other network-connected devices such as mobile devices, scanners, printers, copiers, and fax appliances.

When developed correctly, checklists can greatly assist users in configuring IT products to security baselines that offer more protection than the installed out-of-the-box defaults. Configuring a system to conform to specified security guidance (e.g., NIST Special Publications [SP], Defense Information Systems Agency [DISA] STIGs and checklists) or other security specification is a highly technical task. The following list includes some of the benefits associated with using checklists:

- Providing a baseline level of security to protect against common and dangerous local and remote threats (e.g., viruses and worms, denial of service attacks, unauthorized access, inappropriate usage)
- Automating the verification of technical security controls for assessments and compliance testing

[2] From herein, the Cyber Security Act terminology, *checklist*, will be used to describe a security configuration checklist or what other literature may refer to as a lockdown guide, hardening guide, or benchmark configuration.

- Mapping specific checks to high-level requirements (e.g., NIST SP 800-53, DISA STIG)
- Significantly reducing the time required to research and develop appropriate security configurations for installed IT products by leveraging existing checklists with in-house expertise
- Allowing smaller organizations to leverage outside resources to implement recommended practice security configurations
- Preventing public loss of confidence or embarrassment due to compromise of publicly accessible systems.

While the use of security checklists can significantly improve overall levels of security in organizations, no checklist can permit a system or a product to become 100% secure. However, use of checklists that emphasize hardening of systems against the hidden flaws or bugs inherent in software, including verifying configuration settings and the application of patches, will typically result in greater levels of product security and protection from future threats (e.g., zero-day vulnerabilities).

Many security checklists have already been created for protecting various IT products from threats. Section 1.1 discusses these efforts, and it also addresses other existing efforts related to the standardization and distribution of security checklists. Section 1.2 explains the increased need to have automated checklists (e.g., configuration files and tools to apply them) instead of manually applied checklists (e.g., English prose instructions).

1.1 Existing Efforts

To date, there have been many separate efforts to produce security checklists for IT products. This section describes several of these efforts, as well as complementary efforts to assist in the identification and classification of vulnerabilities in IT products.

1.1.1 Checklist Creation

Checklists have been developed by many parties, including IT vendors, consortia, academia, industry, Federal agencies and other governmental organizations, and others in the public and private sectors. Examples of well-known checklist creation efforts are as follows:

- The Defense Information Systems Agency (DISA) produces two types of documents that have specific recommendations for the Department of Defense (DOD). Security Technical Implementation Guides (STIG) explain security principles for a class of products (e.g., desktop applications, wireless networking) or a specific product, and list high-level requirements for securing the product or class of products. A related set of corresponding documents called checklists provide detailed instructions for evaluating systems to verify that they meet the requirements specified in the STIGs.
- The National Security Agency (NSA) produces a variety of security configuration guides, including guides for classes of products (e.g., routers, IP telephony) and guides for specific products.
- NIST has produced checklists specifically tailored for Federal agency use that are based on checklists produced by DISA, NSA, IT product vendors, and other sources. These checklists consist of publications that describe the settings, and some checklists also include configuration files (specifically, .inf files) that can be applied to particular Microsoft operating systems.
- The Center for Internet Security (CIS) develops checklists that are intended to be used by a wide range of organizations and individuals. CIS checklists are created through a consensus process involving the recommendations of many participants.

- IT product vendors such as Microsoft Corporation, Sun Microsystems, ThreatGuard, Citadel, and Hewlett-Packard have been publishing security guidance documents for many of their products.

1.1.2 Checklist Repository Programs

It can be time-consuming to find checklists for specific products since the checklists are created by many different sources. NIST decided to establish a centralized repository of checklist information to assist checklist users in finding what they need. NIST has a beta version of its Security Configuration Checklists Repository available for public use at http://checklists.nist.gov/. The repository contains checklists that have been developed and screened to meet the requirements of the NIST Security Configuration Checklists Program for IT Products, which is the predecessor to the new National Checklist Program. As of mid-2006, the beta version of the repository hosted over 115 checklists addressing over 155 specific IT products. The format of the checklists varies widely, from documents written in English prose to files and scripts written for use with automated tools. An example of such a tool is the DISA Gold Disk, which can scan a system for configuration settings that differ from policy requirements. Table 1 lists the primary contributors to the beta version of the repository.

Users of the repository can browse checklist descriptions to locate and retrieve a particular checklist using a variety of different fields, including the product category, vendor name, and submitting organization.

Table 1: Primary Contributors to the NIST Checklist Repository Beta

Contributor	Contributor's Web Site for Checklists
CIS	http://cisecurity.org/
DISA	http://iase.disa.mil/stigs/checklist
Hewlett-Packard	http://www.hp.com/
LJK Software	http://www.ljk.com/
Microsoft	http://www.microsoft.com/
NIST	http://checklists.nist.gov/
NSA	http://www.nsa.gov/snac/

1.1.3 Vulnerability Identification and Classification

It is very helpful to have checklists use consistent references for vulnerabilities so that it is apparent which vulnerabilities each checklist addresses. The Common Vulnerabilities and Exposures (CVE) vulnerability naming standard[3] is a dictionary of names for most publicly known security flaws in IT software. The CVE industry standard has achieved wide acceptance by the security industry and a number of government organizations. It is funded by US-CERT and the technical analysis work is done at MITRE Corporation. General CVE information is available at http://cve.mitre.org/.[4]

CVE provides the computer security community with the following:

[3] CVE has not been adopted by any formal standards body. It is a widely used self-declared standard. NIST SP 800-51, *Use of the Common Vulnerabilities and Exposures (CVE) Vulnerability Naming Scheme*, is available at http://csrc.nist.gov/publications/nistpubs/800-51/sp800-51.pdf.

[4] In mid-2006, plans for a Common Configuration Enumeration (CCE) standard were announced. It is very similar to CVE, but it addresses security misconfigurations in software deployments instead of flaws with the software itself. Checklists could refer to CCE names for misconfigurations just as they refer to CVE names for software flaws. More information on CCE is available at http://cve.mitre.org/cce/.

- A comprehensive list of publicly known vulnerabilities
- An analysis of the authenticity of newly published vulnerabilities
- A unique name to be used for each vulnerability.

The vulnerabilities listed in CVE can be best viewed using the National Vulnerability Database (NVD), which provides summaries for all CVE vulnerabilities. Each summary contains attributes of the vulnerability (including a short summary and vulnerable version numbers) and links to advisories, patches, and other resources related to the vulnerability. NVD offers a fine-grained search engine that allows users to search for vulnerabilities containing a variety of characteristics. For example, users can search on product characteristics such as vendor name, product name, and version number, or on vulnerability characteristics such as severity, related exploited range, and type of vulnerability. NVD also supports queries in OVAL format, which is described in Section 1.1.4. NVD is available at http://nvd.nist.gov/.

1.1.4 Vulnerability and Checklist Languages

Checklists can be developed using many different formats; however, having standard formats supports interoperability and ease of use. One language widely used for checklists, Extensible Configuration Checklist Description Format (XCCDF), can define structured collections of security configuration rules for sets of target systems. The XCCDF specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices. More information on XCCDF is available in Sections 3 and 4 of this publication and from NIST Interagency Report (IR) 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF)*.[5]

Another language widely used for checklists, Open Vulnerability and Assessment Language (OVAL), is utilized by security experts to exchange technical details about how to check for the presence of vulnerabilities and configuration issues on computer systems. The vulnerabilities and configuration issues are identified using tests—OVAL definitions in Extensible Markup Language (XML)—that can be utilized by end users or implemented in information security products and services. OVAL provides a standard XML format for vulnerability identification and scan criteria for vulnerabilities. More information on OVAL is available at http://oval.mitre.org/ and in Sections 3 and 4 of this publication.

1.2 The Need for Automated Checklists

It has become increasingly important to have automated methods for applying checklists to systems and comparing checklist settings to the actual settings on systems. Reasons for this include the following:

- Guidance in support of compliance initiatives (e.g., FISMA, STIGs) has become more complex, making it more difficult to determine the applicability of a configuration given the IT system's categorization (e.g., low/moderate/high, Mission Assurance Category [MAC], confidentiality [public, sensitive, secret]).
- Organizations need to secure a much greater number of systems and a much wider variety of software.

[5] NIST IR 7275 is available for download at http://checklists.nist.gov/docs/xccdf-spec-1.1.pdf. The text in this paragraph was derived from NIST IR 7275.

- Software has become more complex, and as a result there are typically many more security-related configuration settings available, as well as more vulnerabilities that need to be mitigated.
- There is considerable cost in customizing security tools to implement policy. The use of automated methods as a starting point allows tool vendors to preconfigure them using checklists based on common policies.
- Systems face many more threats, both in terms of unique threats (e.g., types of attacks) and in the frequency of attack attempts (e.g., malware).
- The increased rates of new vulnerabilities and new threats cause organizations to need to change the security configuration of their systems more frequently.
- Organizations need to verify the security posture of many systems regularly as part of security assessments and security compliance efforts.

Automation is needed to ensure that security settings are applied and verified consistently throughout an organization and also among different organizations. Benefits of automating the use of security checklists include the following:

- Easier to ensure compliance with a single policies or multiple policies (e.g., FISMA, Health Insurance Portability and Accountability Act [HIPAA] of 1996, STIGs)
- Permits faster, more cooperative, and more automated definition of security rules, procedures, guidance documents, alerts, advisories, and remediation measures
- Permits fast, uniform, manageable administration of security checks and audits
- Permits composition of security rules and tests from different community groups and vendors
- Permits scoring, reporting, and tracking of security status and checklist conformance, both over distributed systems and over the same systems across their operational lifetimes
- Fosters development of interoperable community and commercial tools for creating and employing security benchmarks and guidance data.

As of mid-2006, automated security checklists are available for some products, but many of these checklists use different formats or can only be applied using proprietary tools. Also, many checklists address some aspects of security configuration in depth, while addressing other aspects at a higher level only or not at all. To be more effective at securing systems and more efficient to use, automated security checklists need to be more consistent and comprehensive. Fostering the creation of such checklists is the primary goal of the Security Content Automation Program, which is described in Section 2.

2. The Security Content Automation Program

In 2004, NIST launched its Security Configuration Checklists Program for IT Products. The goal for the program was to create a central repository for IT product security checklists of any kind. This would make it much easier for people to find up-to-date security checklists, compare the levels of security that the checklists provided, and determine how the checklists should be implemented. As described in Section 1.1.2, the program has been successful, and the beta repository has been widely used. However, while many of the checklists in the repository are of high quality, they use many different specification, test, and report formats, which limits their usefulness for most audiences. Also, commercial and community developers have created automated tools for applying checklists, but most of these tools use different, often proprietary, data formats for both input (checklists) and output (reports). The Security Configuration Checklists Program was recently renamed the NIST National Checklist Program (NCP).

The Security Content Automation Program (SCAP)[6] supports the goals of the National Checklist Program, and it additionally seeks to encourage the development of checklists that are compliant or compatible with XCCDF and/or OVAL, as described in Section 1.1.4. XCCDF and OVAL are designed to enable easier, more uniform creation of checklists, and to allow them to be used with a variety of commercial, open source, and government off-the-shelf (GOTS) tools. *Security content automation* is the process of using tools, scripts, and other technologies to automate the application or verification of security-related configuration settings for operating systems and applications, as specified in XCCDF and/or OVAL-compliant or compatible checklists. SCAP leverages separate but complementary government efforts as part of its integrated solution for security content automation. The product of the NCP will be a single centralized database that provides both security baseline guidance and information on newly discovered vulnerabilities in the standardized XCCDF and OVAL formats on a per-platform basis that is downloadable by the public.

This section provides an overview of the SCAP. Section 2.1discusses the benefits that the program provides. Section 2.2 explains how the program categorizes each checklist to make it easier for users to find the appropriate checklists for their needs. Sections 2.3 and 2.4 provide overviews of the program's processes for checklist development and usage, respectively.

2.1 Program Benefits

The primary intentions of the SCAP is to improve the application, verification, and reporting of security configuration settings. The benefits of this include the following:

- Strengthening the security of IT systems at Federal agencies and other organizations
- Enabling an automated approach for Federal agencies and other organizations to achieve compliance to legislation such as FISMA, HIPAA, and Sarbanes-Oxley Act of 2002 [SOX]) for the technical security controls on their systems. The program benefits compliance efforts in several ways, such as the following:
  - Ensuring that assessments are performed consistently (e.g., equal coverage, high quality, minimizing false positives/negatives, proper scan disposition)
  - Supporting the traceability of specific security settings to corresponding compliance requirements
  - Providing standardized scan criteria

[6] The preferred pronunciation for SCAP is "ESS-cap".

  – Decreasing the cost of compliance verification

- Encouraging higher accuracy of commercial-off-the-shelf (COTS) and government-off-the-shelf (GOTS) security scanning and remediation products through the standardization of formats, the availability of content, and the propagation of standards and content
- Reducing costs for the IT security industry by eliminating substantial duplication of effort and standardizing vulnerability and misconfiguration checking activities.

Examples of these benefits are presented below.[7]

2.1.1 Streamlining Compliance to Policies

Most organizations have difficulty measuring the security of their IT systems. They also struggle first to implement technical policy (e.g., NIST SPs, DISA STIGs) and then to demonstrate unambiguously to various audiences (e.g, Inspector General, auditors) that they have complied and ultimately improved the security of their systems. This difficulty arises from various causes, such as different interpretations of policy, the complexity of systems, and human error. Another cause is the absence of standard assessment criteria; because of this, vulnerability identification and subsequent policy compliance tools rely on their own interpretation of what constitutes compliance or a vulnerability. In the case of identification, different tools assessing the same vulnerability often generate different results.

The SCAP proposes to automate certain technical aspects of security by making configuration information available in machine-readable XML formats instead of human-readable configuration guides, checklists, databases, etc. The standard XML formats allow organizations to use COTS, GOTS, or open source tools to automatically check their security and map it to technical compliance requirements. Security checklists are of particular interest to Federal agencies in meeting the security requirements of FISMA. FISMA (section 3544(b)(2)(D)(iii)) requires each agency to determine minimally acceptable system configuration requirements and ensure compliance with them. Checklists can also map specific technical control settings to the corresponding NIST SP 800-53 controls, which can make the verification of compliance more consistent and efficient. For example, a checklist could examine the password strength settings on a system and report whether or not those settings meet the requirements specified in NIST SP 800-53.

The development and sharing of these checklists can greatly reduce what would otherwise be a “reinvention of the wheel” for IT products that are widely used in the Federal government, such as common operating systems, servers, and client applications. Currently, redundant efforts are assumed by Federal agencies in identifying and categorizing vulnerabilities into groupings to satisfy compliance regulations. At the heart of this problem is the lack of standardized assessment data and the format to facilitate the communication or propagation of such data.

IT vendors that provide recommended checklists “out of the box” that adhere with the FISMA-associated security control baselines will not only provide more consistency in configuration settings within the Federal agencies but also provide a much more cost-effective method of establishing and verifying the minimum configuration settings, even if the agencies modify the original checklists provided by checklist developers to fine-tune the configuration settings for their particular applications and operational environments.

[7] The examples are based on scenarios presented in Section 1.2 of NISTIR 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF).*

2.1.2 Combining and Customizing Checklists

An organization may wish to use settings from multiple checklists in creating organization-specific checklists. For example, an organization might want to use settings from a checklist created by an academic group and add to them supplementary settings from a separate checklist created by a government agency. The SCAP makes this simple by promoting the use of standard formats for the checklist and standard identifiers for vulnerabilities. An organization can easily reuse material from one checklist in other checklists and customize the resulting checklist as needed to correspond to the organization's internal security policy. As long as the resulting checklist complies with the standard formats, it can be used with existing tools to perform assessments and audits, and the tools can generate reports that specify remediation measures that will bring the evaluated systems into full internal policy compliance. Another example of combining checklists is having a checklist refer to one or more other checklists, such as a checklist for a database product referencing the checklist for the operating system on which the database product runs.

An organization can easily compare multiple checklists if they are in compatible automated formats. For example, there are tools that can load and display settings from multiple checklists simultaneously, and even allow users to select which portions of each checklist should be used. This is much easier than manually comparing several sets of checklists in different formats, such as English prose, spreadsheets, and scripts. It is particularly helpful for determining compliance with multiple compliance efforts, such as NIST SP 800-53 and DISA STIGs.

2.1.3 Quickly Distributing Assessment Instructions

Having standard formats for checklists allows organizations to easily share information on assessing new vulnerabilities. For example, a Federally-funded laboratory that issues a security advisory about a new Internet worm would not only provide a prose description of the worm's attack vector, but it would also create and make available a set of configuration settings in a standard format that allow organizations to assess their systems' vulnerability to the worm. Organizations all over the world would read the advisory, and then use installed tools that support the standard format to check their status and fix vulnerable systems.

2.2 Checklist Categorization

As discussed earlier in the publication, there are different types of checklists. The SCAP categorizes checklists based on their level of automation:

- **Automated.** An *automated checklist* is one that is applied through one or more tools that automatically alter or verify settings based on the contents of the checklist. For example, many checklists are written in the Extensible Markup Language (XML), and special tools exist that can use the contents of the XML files to check and alter settings on systems.
- **Non-Automated.** As the name implies, a *non-automated checklist* is one that is designed to be implemented manually, such as English prose instructions that describe the steps that an administrator should perform to secure a system or verify its security settings.

For both automated and non-automated checklists, the SCAP also categorizes each checklist into one of three tiers, as follows:

- **Tier 1.** The checklist provides helpful guidance on securing a specific product. However, the checklist is considered ready for use on Federal agency systems after first performing extensive modifications and testing. The checklist has not necessarily been tested in production environments.

Also, the vendor of the product secured by the checklist does not necessarily support or endorse the use of the checklist; in some cases, using the checklist could possibly violate the terms of a support contract with the product's vendor.

- **Tier 2.** The checklist is considered ready for use on Federal agency systems after first performing moderate modifications and testing. The controls in the checklist are mapped to the corresponding controls from NIST SP 800-53. The checklist has been tested in production or mirrored production environments. The vendor of the product secured by the checklist will continue to provide support for systems to which the checklist is applied.
- **Tier 3.** The checklist is specific to Federal agencies,[8] so it is considered ready for use on Federal agency systems after performing minor adjustments and testing to address organization-specific policies and environmental differences. The controls in the checklist are mapped to the corresponding controls from NIST SP 800-53. The checklist has been tested in production or mirrored production environments. The vendor of the product secured by the checklist will continue to provide support for systems to which the checklist is applied. Also, if the checklist is automated, it is OVAL or XCCDF-compliant or compatible; these terms are defined below.

Table 2 compares the six checklist categories (automated and non-automated, tiers 1 through 3). The following items explain the characteristics listed in the table:

- "Maintenance commitment" means that the checklist submitter has agreed to maintain the checklist.
- "Vendor supported" means that if the vendor provides support for a specific product, that the vendor will continue to provide support for the product after the checklist has been used to configure it.
- "Operationally tested" means that the content has been used on production or mirror systems in a production or mirror environment and the results reviewed to identify potential problems with the recommendations.
- "FISMA mapping" means that the checklist maps its specific security control settings to the corresponding controls from NIST SP 800-53.
- "Federal agency specific" means that the content is tailored to meet general Federal agency requirements. Examples include:
  - Requiring the use of FIPS-compliant encryption
  - Mapping against FIPS 199 low/moderate/high categories
  - Mapping against NIST SP 800-70 environment categories (enterprise, SOHO, etc.)
- "OVAL and XCCDF compliant" means that the content is in validated OVAL and XCCDF format. "OVAL and XCCDF compatible" means that the content is provided in a format that can automatically be translated to OVAL and XCCDF. For example, content could be provided in a database, and a database query could allow the content to be extracted and formatted in an OVAL and XCCDF-compliant manner.

[8] This requires that the checklist's settings or instructions support Federal agency-specific requirements such as the use of Federal Information Processing Standards (FIPS) compliant cryptographic algorithms. Additionally, when applicable, the checklist should also support Federal agency compliance efforts, such as FISMA, by mapping compliance requirements to specific settings or instructions.

Table 2: Comparison of Checklist Categories

Checklist Category	Maintenance Commitment	Vendor Supported	Operationally Tested	FISMA Mapping	Federal Agency Specific	OVAL and XCCDF Compliant or Compatible
Automated Checklists						
Tier 1	Required	Optional	Optional	Optional	Optional	Optional
Tier 2	Required	Required	Required	Required	Optional	Optional
Tier 3	Required	Required	Required	Required	Required	Required
Non-Automated Checklists						
Tier 1	Required	Optional	Optional	Optional	Optional	N/A
Tier 2	Required	Required	Required	Required	Optional	N/A
Tier 3	Required	Required	Required	Required	Required	N/A

The checklist categories should be helpful to users in selecting the checklists that best meet their needs. The checklist usage process, including checklist review and selection, is described in Section 2.4. Accordingly, checklist developers should keep the categories in mind when creating each checklist and ensure that the checklist maps to the desired categories. Section 2.3 describes the checklist development process.

2.3 Checklist Development Process

For checklist developers, the development process is composed of two stages. The first stage involves only developer actions, whereas the second stage involves interactions between NIST, the developer, and public reviewers. The first stage contains four steps, as shown in Figure 1:

- In step one, the developer becomes familiar with the procedures and requirements of the checklist program and completes an agreement to participate in the program. Copies of the procedures, requirements, and participation agreement are maintained in NIST SP 800-70, *Security Configuration Checklists Program for IT Products—Guidance for Checklists Users and Developers*.
- In step two, the developer creates, tests, and refines the checklist. For Tier 2 or Tier 3 consideration, the checklist must map its tests to FISMA (NIST SP 800-53) controls, and checklist testing must include production or mirrored production environments. For Tier 3 consideration, the checklist must be specific to Federal agencies, as described in Section 2.2. Automated tools are available for some checklist formats to assist developers, such as verifying checklist syntax.
- In step three, the developer documents the checklist according to the guidelines of the program.
- In step four, the developer prepares a checklist submission package and submits it to NIST.

More information on steps three and four is also available from NIST SP 800-70.

In stage two, NIST then performs the remaining four steps, with interaction from the developer and public reviewers:

- In step five, NIST screens the checklist according to program requirements, including reviewing the FISMA mapping for Tier 2 and 3 checklists, and addresses any issues with the developer.
- The next step is a public review of the checklist, which typically lasts 30 to 60 days. Comments submitted during the review are addressed as applicable by the developer and NIST.

- For step seven, NIST posts the checklist on the repository and announces its presence.
- Lastly, step eight involves periodic updates to the checklist and issues of checklist archival.

NIST SP 800-70 contains a more detailed explanation of the process for screening and publishing checklists.

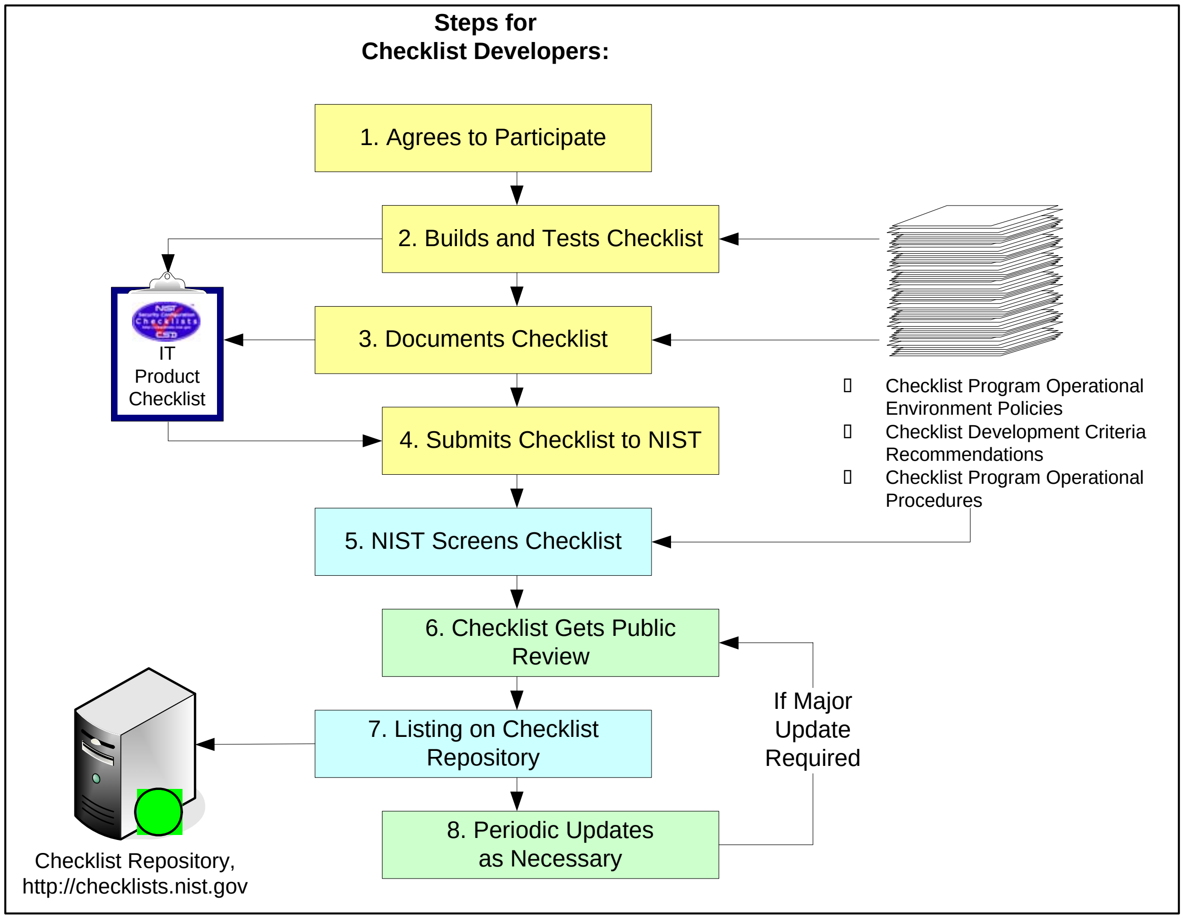
Steps for
Checklist Developers:
1. Agrees to Participate
2. Builds and Tests Checklist
3. Documents Checklist
4. Submits Checklist to NIST
5. NIST Screens Checklist
6. Checklist Gets Public Review
7. Listing on Checklist Repository
8. Periodic Updates as Necessary
IT Product Checklist
Checklist Program Operational Environment Policies
Checklist Development Criteria Recommendations
Checklist Program Operational Procedures
If Major Update Required
Checklist Repository, http://checklists.nist.gov

Figure 1: Steps for Checklist Developers

2.4 Checklist Usage Process

The general steps involved for checklist users are shown in Figure 2. For checklist users, the steps are simple and straightforward:

- In step one, users gather their local requirements (e.g., IT products, the operating environment and associated security needs) and then acquire or purchase the IT product that best suits their needs.
- In step two, users browse the checklist repository to retrieve checklists that match the user's operational environment and security requirements. The repository provides detailed information on each checklist; NIST SP 800-70 describes these checklist information fields. If a product is intended to be secure out-of-the-box (e.g., it was secured by the vendor using a security checklist), it is still

important to check the repository for updates to that checklist. Users should verify the authenticity and integrity of each retrieved checklist before using it (e.g., checking message digests or cryptographic hashes for the checklist files).

- Step three involves tailoring and documenting the checklist as necessary to take into account local security policies and functional requirements, testing the checklist, and providing any feedback to NIST and the checklist developers. For checklists that will be used to modify the security configurations of systems, checklist users should be particularly careful and thorough in their planning and testing.
- Lastly, step four involves preparation for deploying the checklist, such as making configuration or data backups, and then applying the checklist in production, either to configure the product or system to the baseline level of security implemented in the checklist, or to verify that the product or system is already configured properly.

For some checklist formats, tools are available to assist users in performing steps three and four, such as customizing which checks are performed, applying the checklists, recording results, and generating reports.

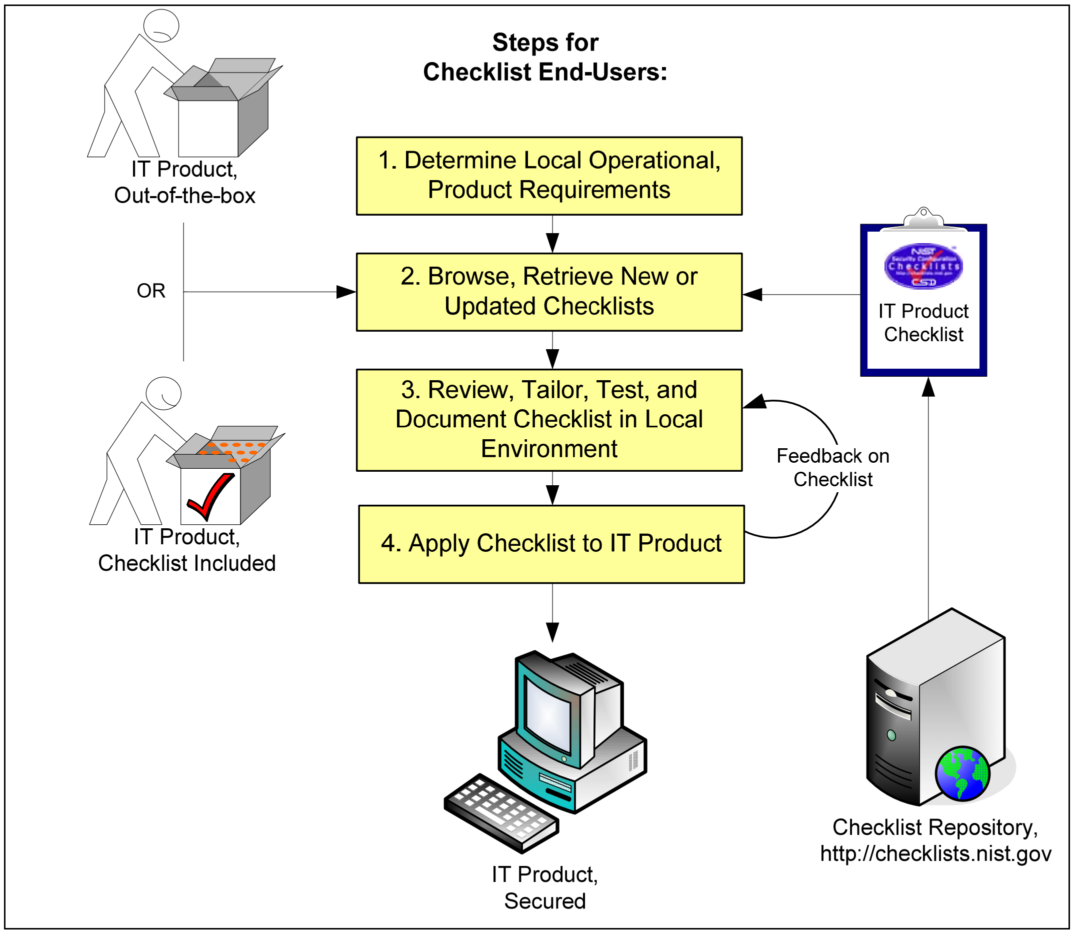
Steps for
Checklist End-Users:
IT Product,
Out-of-the-box
OR
IT Product,
Checklist Included
1. Determine Local Operational, Product Requirements
2. Browse, Retrieve New or Updated Checklists
3. Review, Tailor, Test, and Document Checklist in Local Environment
4. Apply Checklist to IT Product
Feedback on Checklist
IT Product Checklist
IT Product,
Secured
Checklist Repository,
http://checklists.nist.gov

Figure 2: Steps for Checklist Users

NIST SP 800-70 contains more details on considerations associated with each step.

3. Understanding SCAP-Provided XML Content

One of the primary goals of the SCAP is to encourage the development of checklists in XML formats, particularly checklists that are compliant with XCCDF and/or OVAL. This section first provides a high-level overview of XCCDF and/or OVAL, and explains how they can be used for checklists. It then focuses on the use of XCCDF and/or OVAL-compliant checklists for helping agencies with FISMA compliance efforts, and also compares the FISMA and DOD 8500.2/8510 compliance efforts.

3.1 The Basics of XCCDF

As described in Section 1, Extensible Configuration Checklist Description Format (XCCDF) is a specification language for writing security checklists, benchmarks, and related kinds of documents. An XCCDF document represents a structured collection of security configuration rules for some set of target systems. The specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices. Development of the XCCDF specification is being led by NSA, with contributions from other agencies and organizations.[9]

An XCCDF document is composed of one or more XCCDF rules. An *XCCDF rule* is a high-level definition of a technical check on a system. A rule does not directly specify how a check should be performed, but instead points to other XML documents (such as OVAL definition files, which are explained in Section 3.2) that contain the actual instructions for performing the check. Table 3 shows sample values from an XCCDF rule. This particular rule is for ensuring that the minimum password length is set to at least 8 characters. The System Check section of the rule specifies the OVAL definition example presented in Section 3.2.

Table 3: Sample Values from an XCCDF Rule

Rule Field	Sample Data	Explanation
Rule ID	MinimumPasswordLength-8	The identifier for this rule
Title	Minimum Password Length = 8	The title for the rule
Description	This setting specifies the minimum length of a password in characters. The rationale behind this setting is that longer passwords are more difficult to guess and crack than shorter passwords. The downside is that longer passwords are often more difficult for users to remember. Organizations that want to set a relatively large minimum password length should encourage their users to use passphrases, which may be easier to remember than conventional passwords	The description of the rule

9 More information on XCCDF is available at http://checklists.nist.gov/xccdf.html and from NIST IR 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF)*, which is available for download at http://checklists.nist.gov/docs/xccdf-spec-1.1.pdf.

Rule Field	Sample Data	Explanation
References	IA-5 (http://csrc.nist.gov/publications/nistpubs/800-53/SP800-53.pdf)	References to checklists and other documents that contain requirements to which this rule maps—in this case, the IA-5 (Authenticator Management) control from NIST SP 800-53
Requires	IA-5	The group to which this rule belongs, if any; in this case, the IA-5 group
System Check		
Schema	http://oval.mitre.org/OVAL/XMLSchema/oval	Which XML schema should be used; in most cases (including this one), the OVAL schema is specified
OVAL definition file reference	WindowsXP-SP800-68.xml	Name of the OVAL definition file
OVAL definition ID	oval:gov.nist.1:def:20	The identifier of the OVAL definition to be used; in this case, the definition used as the basis for the Table 5 example

An XCCDF document typically contains dozens or hundreds of XCCDF rules. The document also defines at least one *XCCDF profile,* which specifies which rules should be used to check a particular type of system. For example, a profile for a standard enterprise desktop computer could enable a rule that checks for a minimum password length of 8 characters, while a profile for a computer with high security requirements could instead enable a rule that checks for a minimum password length of 12 characters. Each profile can be thought of as a policy that specifies the technical security control setting requirements for a type of system. By creating a policy that corresponds to a particular set of requirements, such as FISMA, STIGs, or HIPAA, the policy can be used to map those high-level requirements to the corresponding OVAL definitions and tests, as described in Section 3.2.

Optionally, an XCCDF document can have one or more *XCCDF groups*; each group contains one or more XCCDF rules and/or one or more other XCCDF groups. Having a group allows multiple related rules to be enabled or disabled collectively instead of individually.

Another option involving XCCDF rules is to have user-definable values for certain rules, known as *XCCDF values*. Table 4 shows sample values from an XCCDF value. This particular value is for setting the duration of an automated account lockout when too many consecutive failed login attempts have occurred. In this case, the value has been set to 15 minutes. To take organization policies specifying a different value into account, a checklist user can either alter this value or override this value with a different number in the profile(s) that have rules that use the value. Both of these options are explained in more detail in Section 4. The Lower-Bound and Upper-Bound fields restrict what numbers the checklist user can enter when specifying a new number for AccountLockoutDurationTime.

Table 4: Sample Values from an XCCDF Value Statement

Rule Field	Sample Data	Explanation
Value ID	AccountLockoutDurationTime	The identifier for this value
Type	number	The type of the value (e.g., string, number, Boolean)
Operator	greater than or equal	The comparison operator (in this case, the system's value for account lockout duration time must be greater than or equal to the specified value)

Rule Field	Sample Data	Explanation
Title	Account Lockout Duration Time	The title for the value
Description	This value specifies how long the user account should be locked out. This is often set to a low but substantial value (e.g., 15 minutes), for two reasons. First, a legitimate user that is accidentally locked out only has to wait 15 minutes to regain access, instead of asking an administrator to unlock the account. Second, an attacker who is guessing passwords using brute force methods will only be able to try a small number of passwords at a time, then wait 15 minutes before trying any more. This greatly reduces the chances that the brute force attack will be successful.	The description of the value
Question	Account lockout duration time (in minutes)	Explanatory text that can be presented to the user when is customizing the checklist
Value	15	The value assigned to the AccountLockoutDurationTime value (called a “value number” below for clarity)
Default	15	A suggested default value number for checklist users’ reference; not actually used when performing checks or applying configuration settings
Lower-Bound	10	If a checklist user alters the value number, it cannot be set lower than 10
Upper-Bound	30	If a checklist user alters the value number, it cannot be set higher than 30

When an XCCDF profile is used to evaluate the security configuration of a system against a policy, the following types of documents are typically generated:[10]

- In the XCCDF language, a *benchmark report* is a human-readable document containing the results of the evaluation, including a list of the rules for which the system passed and failed, and also a “compliance score” that summarizes how closely the system complied with the policy. Many scoring models can be used, but the XCCDF specification describes three:
  - The default scoring model generates a compliance score for each group and also a score at the top level, which is based on the scores for the rules and groups directly referenced by the evaluated profile. The advantage of this model is that it is easy to see which groups of checks were least compliant. The disadvantage of this model is that it is complex—weightings typically need to be specified for the groups and sometimes rules as well to ensure that the relative importance of each group and rule is taken into account when calculating the scores.
  - The flat model generates a single score for all rules that were used, based on the relative weighting assigned to each rule. It does not do any scoring based on group definitions. In the flat model, the weightings of all the selected rules are added together to create a maximum possible score. That score can then be compared to the total of the weightings of all the passed rules. This model is simpler than the default scoring model, but because it will

[10] This material is based on Section 3.3 of NIST IR 7275.

generate different maximum possible scores on different types of systems, it makes it more difficult to compare the relative security of different systems.

  - The flat unweighted model is similar to the flat model, except that it ignores all weightings. This is a very simple model, but the loss of weighting means that a relatively unimportant rule and a critically important rule are counted equally, which could lead to somewhat misleading scores.

- A *benchmark results* file is machine-readable, typically in XML, and contains detailed information on the results of the evaluation.
- A *fix script* is a machine-readable file that can be applied to a system to fix some or all of the problems that were identified during the evaluation.

3.2 The Basics of OVAL

As described in Section 1, the *Open Vulnerability and Assessment Language (OVAL)* is used to specify the technical details for checking systems for the presence of vulnerabilities and configuration issues. A set of instructions used to check for a security problem, such as an incorrect minimum password length setting, is known as an *OVAL definition*. A file containing one or more OVAL definitions (often hundreds or even thousands) is known as an *OVAL definition file*. A single definition file often contains many more tests than would ever be run against a single system; for example, a file could contain checks for minimum password lengths of at least 8 characters and at least 12 characters, but typically at most one of these two checks would be run against a particular system. Actually, the intention of the SCAP is not to have OVAL definition files used directly to perform checks on systems, but rather to have an XCCDF file use just the OVAL definitions that are needed to check a particular system. Further explanation of this is provided in Section 3.1.

There are four types of OVAL definitions:[11]

- Vulnerability definitions, which define “the conditions that must exist on a computer for a specific vulnerability to be present”
- Patch definitions, which define “the conditions on a computer that determine whether a particular patch is appropriate for a system”
- Inventory definitions, which define “the conditions on a computer that determine whether a specific piece of software is installed on the system”
- Compliance definitions, which define “the conditions on a computer that determine compliance with a specific policy or configuration statement”.

Table 5 shows sample values that have been extracted from an actual OVAL compliance definition. Explanations of each value have also been provided. The definition ID, version, and class are standard fields that are part of every OVAL definition. The exact types of information contained in the metadata vary among definitions, but at a high level they explain the intent of the definition. The criteria provide the technical details of how the system will be checked for the items of interest, such as the presence of a vulnerability or the value of a configuration setting. Every OVAL definition has one or more criteria, and the definition specifies how the results produced by the criteria are combined (e.g., AND, OR).

[11] These definitions are taken from the OVAL Web site’s “Structure of the Language” page, located at http://oval.mitre.org/language/about/structure.html.

The example in Table 5 has two criteria. One of the criteria is an *OVAL test*, which is a specific system check—in this case, that the system is configured to require a minimum password length of at least 8 characters. The other criterion is actually another definition—in this case, an inventory definition that confirms that the target system is running Windows XP SP2 on a 32-bit architecture. Because of the order of the criteria, the inventory definition will be checked before the test is performed, which makes sense because the test may not be valid if run on a different operating system version.

Table 5: Sample Values from an OVAL Criteria Definition

Definition Field	Sample Data	Explanation
Definition ID	oval:gov.nist.1:def:20	The identifier for this definition; unique within the OVAL definition file
Definition version	1	The version of the definition
Definition class	compliance	Which type of definition this is
Metadata		
Title	Minimum Password Length of 8 Characters	The title for the definition
Affected product	Microsoft Windows XP, SP2, 32 bit	The operating system or application version(s) to which this definition is applicable
References	NIST SP800-68 Appendix A, 1.4b, http://csrc.nist.gov/itsec/download_WinXP.html DISA FSO Checklist, 5.4.1.3 DISA VMS 6XID V0001106 DISA PDI ID 1740	References to checklists and other documents that contain requirements to which this definition maps
Description	Minimum password length is 8 characters	The description for the definition—often considerably longer than the title, but in this case roughly the same length
NIST	IA-5	Specific security controls from NIST SP 800-53 to which this definition maps—in this case, IA-5, which is Authenticator Management[12]
Additional references	NSA NT Guide: Chap 5, p. 30; NSA WIN2K Guide, Group Policy: Security Configuration Toolset: Chap. 3, p. 22; NSA XP Guide: Chap. 4, p. 21; DODD 8500.1 Para 4.18; DODI 8500.2 DCCS-2, DCSC-1; CJCSM 6510.01 App. A, Enclosure A, Para. 5.b (8)	References to additional security configuration approaches that contain requirements to which this definition maps
Criteria		
Definition reference	oval:gov.nist.1:def:9	The identifier of another OVAL definition, which is evaluated to determine if it is true or false. In this case, the referenced definition calls four tests, and if they are all evaluated as true, then the definition is true, otherwise it is false.
Definition comment	Precondition 9: Windows family, Windows XP, SP2, 32 bit	A brief explanation of what the definition addresses; in this case, it is used to determine if the target system is running Windows XP SP2 on a 32-bit architecture

[12] The XCCDF profile that references this OVAL definition also specifies IA-5 as the NIST SP 800-53 mapping. This has been done in both the XCCDF document and OVAL definition file in case either file is used without the other.

Definition Field	Sample Data	Explanation
Test reference	oval:gov.nist.1:tst:16	The identifier of an OVAL test, which is evaluated as true or false as part of evaluating this definition
Test comment	Minimum password length is 8 characters	A brief explanation of what the test addresses; in this case, it is used to determine if the target system requires a minimum password length of 8 characters

As the example in Table 5 shows, definitions often reference one or more tests. The instructions that comprise each test are also included in the OVAL definition file. A test does not directly contain the technical details of checking the system, but instead references other OVAL constructs. Typically a test references an *OVAL object*, which is a logical construct for a portion of the target system (e.g., password policy, file, Windows registry key), and an *OVAL state*, which is a particular check of the specified OVAL object (e.g., verifying that the password policy requires a minimum password length of at least 8 characters, verifying the existence of a file). An OVAL state can also reference one or more OVAL variables, which are user-definable values (e.g., minimum password length value of 8). This might sound unnecessarily complex, but it is a highly modular approach that greatly reduces redundancy and allows people to use the OVAL definitions without having to understand the details behind them, such as how the tests are constructed. For those people that want all the details, such as checklist developers, the information is easily accessed by searching the OVAL definition file for the definition, test, object, and state ID numbers, and reading the instructions associated with each entity. More technical details on OVAL definition files, including examples of the XML code for OVAL definitions, are presented in Section 4. An OVAL definition tutorial is also available from the OVAL Web site at http://oval.mitre.org/language/about/definition.html.

3.3 Using XCCDF and OVAL for FISMA Compliance

Checklists intended for use in the Federal government are more valuable if they map to the FISMA security control baselines. NIST SP 800-53, *Recommended Security Controls for Federal Information Systems*, provides a catalog of security controls for FISMA compliance. It uses groupings of the controls to create three minimum baseline security control sets for Federal information systems—low, moderate, and high impact, as specified in Federal Information Processing Standards (FIPS) Publication (PUB) 199, *Standards for Security Categorization of Federal Information and Information Systems.*[13] Every system needs to be protected, but the level of protection may vary based on the value of the system and its data; low, moderate, or high impact estimates the potential impact of a security breach involving that particular system. Accordingly, FISMA specifies the most stringent minimum security controls for high impact systems and the least stringent for low impact systems.

To support FISMA compliance, an XCCDF document could contain separate policies for low, moderate, and/or high impact systems (in many cases, a system is unlikely to be used at all three impact levels, such as an enterprise firewall not being low impact). Checklist users can then use the same XCCDF document and associated OVAL definition files to assess similar systems that are at different impact levels, which is much more convenient and efficient than having separate documents and files for each impact level.

Another way to tailor XCCDF documents to support FISMA compliance is to have them take into account the different generic operational environments in which systems function. For example, a system located in a secured agency-owned building and connected to a protected internal network might have different security needs than a similar system used on an employee's home network or directly connected

[13] FIPS PUB 199 is available for download from http://csrc.nist.gov/publications/fips/index.html.

to the Internet. Having XCCDF profiles take these major environmental differences into account can be very helpful to the checklists' users by reducing the amount of time they need to customize the checklists for their systems' environments.

The SCAP identifies the following broad and specialized operational environments, any one of which should be common to most audiences:

- **Managed** or **Enterprise** are typically large organizational systems with defined, organized suites of hardware and software configurations, usually consisting of centrally-managed workstations and servers protected from the Internet by firewalls and other network security devices.
- **Standalone** or **Small Office/Home Office** (SOHO) describes small, informal computer installations that are used for home or business purposes. Standalone encompasses a variety of small-scale environments and devices, ranging from laptops, mobile devices, or home computers, to telecommuting systems, to small businesses and small branch offices of a company.
- **Custom** environments contain systems in which the functionality and degree of security do not fit the other environments. Two typical Custom environments are **Specialized Security-Limited Functionality** and **Legacy**:
  - **Specialized Security-Limited Functionality (SSLF).** An SSLF environment contains systems and networks at high risk of attack or data exposure, with security taking precedence over functionality. It assumes systems have limited or specialized (not general purpose workstations or systems) functionality in a highly threatened environment such as an outward facing firewall or public Web server or whose data content or mission purpose is of such value that aggressive trade-offs in favor of security outweigh the potential negative consequences to other useful system attributes such as legacy applications or interoperability with other systems. Checklists for this environment are not recommended for home users or for large scale general purpose systems. An SSLF environment could be a subset of another environment.
  - **Legacy.** A Legacy environment contains older systems or applications that may use older, less-secure communication mechanisms. Other machines operating in a Legacy environment may need less restrictive security settings so that they can communicate with legacy systems and applications. A Legacy environment could be a subset of a Standalone or Managed environment.

Separate XCCDF profiles can be created for each applicable operational environment in which a system might be deployed. However, it is more helpful to create profiles that take into account both the three impact levels and the four operational environments. This means that for a particular type of target, the XCCDF document could contain up to 12 different profiles.   In most cases, not all profiles will be needed because the target is not expected to be used in certain environments or assigned certain impact levels, or the target is not expected to have certain impact/environment combinations. For example, an enterprise intrusion detection and prevention system would not be run in a SOHO environment, and it would not have an impact level of low. Another example is that SSLF environment would not have any low impact systems.

Table 6 shows an example of how the minimum password length requirement for a Windows XP Professional system might vary based on impact level and operational environment. The N/A entries reflect impact/environment combinations that are not deemed feasible, so the XCCDF document for this target system would contain 10 profiles. For those profiles that show a value of 8 in Table 6, the profile would use the MinimumPasswordLength-8 rule (the rule shown in Table 3); for the entries with a value of

12, the profile would use the MinimumPasswordLength-12 rule. Table 7 provides another view of how these two rules are used by the 10 profiles.

Table 6: Example of Minimum Password Lengths by Impact and Environment

Environment	High	Moderate	Low
Enterprise	12	8	8
SOHO	12	8	8
SSLF	12	N/A	N/A
Legacy	12	8	8

The 10 profiles would each use a somewhat different combination of rules to specify the requirements imposed by the different impact levels and operational environments. Table 7 illustrates how several sample rules might be used by the profiles. Most of the sample rules, such as those for password history enforcement and account lockout reset, are used by all 10 of the profiles. Other rules are used by only some profiles, including the password length rules and the account lockout threshold rules.

Table 7: Examples of Rule Usage for Windows XP Professional Profiles

Rule Identifier	Rule's SP 800-53 Control Mapping	Enterprise/High	Enterprise/Moderate	Enterprise/Low	SOHO/High	SOHO/Moderate	SOHO/Low	SSLF/High	Legacy/High	Legacy/Moderate	Legacy/Low
PasswordHistoryEnforcement	IA-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MaximumPasswordAge	IA-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MinimumPasswordAge	IA-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MinimumPasswordLength-8	IA-5		✓	✓		✓	✓			✓	✓
MinimumPasswordLength-12	IA-5	✓			✓			✓	✓		
PasswordComplexity	IA-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PasswordStorageReversibleEncryption	IA-5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AccountLockoutDuration	AC-7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AccountLockoutThreshold-10	AC-7	✓			✓			✓	✓		
AccountLockoutThreshold-50	AC-7		✓	✓		✓	✓			✓	✓
AccountLockoutReset	AC-7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 7 also illustrates how FISMA requirements can be mapped to specific technical checks. The second column in Table 7 lists the SP 800-53 control to which each rule maps. These controls can be specified by creating a group for each SP 800-53 control (e.g., IA-5, AC-7) and making each rule a member of the appropriate group. This makes it easy to determine which SP 800-53 controls a particular profile partially or fully checks, and it allows scores to be produced for each defined control. If the groups for the controls within a family (e.g., AC-1, AC-2, AC-3) are also placed into a separate group (e.g., AC), then scores can be generated for each family, as well as each individual control.

3.4 Comparing FISMA and DOD 8500.2/8510 Compliance

Like most Federal agencies, both NIST and DISA have the operational requirement of performing FISMA-compliant reporting, which assumes a granular asset inventory (both computing and non-computing), asset posture, and asset security as a function of actual and potential vulnerabilities and security control compliance. Both NIST and DISA are also tasked via legislation or guidance documents (FISMA and DOD 8500.2/8510, respectively), to provide guidance for security compliance processes to their respective audiences. NIST does this in the form of NIST SP 800-53 and related documents (e.g., FIPS 199), and DISA does this through their STIGs and associated technology-specific checklists.

In meeting their similar objectives for different audiences, both NIST and DISA need to define how a system can be categorized as a function of mission criticality. Associating the appropriate vulnerabilities with a system is a daunting task in the absence of automated tools and agreed-upon characterizations. Both agencies have provided guidance for determining the following:

- What constitutes an asset (both computing and non-computing)
- The category of the asset and the controls that must protect the asset
- Appropriately applicable vulnerabilities (AAV)
- Status of an asset as a function of the AAV.

Making these determinations for systems, such as part of certification and accreditation (C&A) efforts, can be quite complex.

NIST and DISA have developed separate, yet similar, approaches to determining the category of an asset:

- NIST SP 800-53 defines a methodology based on assigning a high, moderate, or low rating to the potential impact of disruptions of confidentiality, integrity, and availability of a system.
- DOD 8500.2/8510 defines a methodology that focuses strictly on the potential impact of disruptions of confidentiality. It assigns an impact rating of Mission Assurance Category (MAC) 1, 2, or 3. It also categorizes systems as public, sensitive, and classified.

Given the similarity of their missions for different sectors, NIST and DISA have and will continue to share tools and information for the benefit of their respective audiences. Also, FISMA and DOD 8500.2/8510 require many of the same controls; Appendix G of NIST SP 800-53, Revision 1 maps controls between the two initiatives to demonstrate that the majority of SP 800-53 controls correspond to one or more controls from DOD 8500.2,with slight differences. Therefore, many checklist components, such as XCCDF rules and OVAL criteria and tests, could potentially be used for both FISMA and DOD 8500.2 compliance efforts, as long as the components are mapped correctly to the corresponding FISMA and DOD 8500.2 requirements.

4. Customizing SCAP-Provided XML Content

Checklist users often have to customize checklists to meet their organizations' needs. One common reason is to ensure that a checklist takes into account an organization's specific policies and environment characteristics. Another is to take a checklist for a general type of system (e.g., any system running a particular OS type and version) and customize it for such a system in a particular role, such as an e-mail server or Web server. In many cases, an organization can customize checklists to meet their needs with just a few simple modifications. This section provides an introduction into customizing XCCDF and OVAL-based checklists provided through the NIST SCAP.

4.1 XCCDF Customization

Because XCCDF documents essentially contain the policy for the target system, and most customizations to checklists involve altering the default policy to correspond to the organization's policies and environment, most alterations of SCAP-provided XML content involve changing XCCDF documents. This section focuses on the minor adjustments most often performed to customize XCCDF content. It is outside the scope of this publication to explain how to make major modifications to checklists or to create a new XCCDF document from scratch; information supporting those tasks is available from NISTIR 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF).* There are also many options available for XCCDF statements that are not discussed in this publication, but NISTIR 7275 contains explanations of them as well.

This section uses a top-down approach to explaining the checklist customization process: from profiles and groups to rules and values.

4.1.1 XCCDF Profiles

As mentioned previously, an XCCDF profile is essentially a policy that is applied to the target system or compared to the configuration of the target system. Figure 3 shows an example of some of the XML code that comprises a profile. Because profiles can specify the rules and groups of rules that should be used, profiles are often hundreds or even thousands of lines long. This excerpt shows only a sampling of the code for brevity. The following briefly explains what the excerpt contains:

- The first line assigns a profile ID and the second line gives a title for the profile.
- The next set of lines specifies that several groups (e.g., AC-9, AC-10, AC-15) be deselected for this profile. By default, all groups are selected,[14] so these settings override the defaults and ensure that the rules in those groups will not be used.
- After the Password Policy Settings comment, the next set of lines specifies which rules are selected or deselected for this profile. For example, the PasswordHistoryEnforcement rule will be used, but the MinimumPasswordLength-12 rule will not be used.
- The next-to-last line defines a value that will be used by one of the rules. In this case, it is defining the account lockout duration time as 30 minutes. This value happens to be used by the rule called in the preceding line, AccountLockoutDuration.

[14] The definition of a group can include a different default setting, so a particular group could be deselected by default. However, in most cases this is not done because it is much clearer to have all the group selected/deselected settings in one place—the profile—instead of scattering them throughout a long XCCDF document. Also, if the XCCDF document has multiple profiles, it is extremely helpful to have all of the selected/deselected settings in a single place to facilitate comparisons of the profiles.

- The last line indicates the end of the profile definition.

```
<cdf:Profile id="Enterprise-Moderate">
   <cdf:title>Enterprise-Moderate</cdf:title>
   <cdf:select idref="AC-9" selected="0" />
   <cdf:select idref="AC-10" selected="0" />
   <cdf:select idref="AC-15" selected="0" />
   <cdf:select idref="AC-16" selected="0" />
   <cdf:select idref="AT-5" selected="0" />
   <cdf:select idref="AU-10" selected="0" />
   <cdf:select idref="PE-4" selected="0" />
   <!--   *****************************   -->
   <!--   Password Policy Settings   -->
   <!--   *****************************   -->
   <cdf:select idref="PasswordHistoryEnforcement" selected="1" />
   <cdf:select idref="MaximumPasswordAge" selected="1" />
   <cdf:select idref="MinimumPasswordAge" selected="1" />
   <cdf:select idref="MinimumPasswordLength-8" selected="1" />
   <cdf:select idref="MinimumPasswordLength-12" selected="0" />
   <cdf:select idref="PasswordComplexity" selected="1" />
   <cdf:select idref="AccountLockoutDuration" selected="1" />
   <cdf:set-value idref="AccountLockoutDurationTime">30</cdf:set-value>
</cdf:Profile>
```

Figure 3: Example of XCCDF Profile Definition

This profile example could be customized in a few ways, as follows:

- **Changing which groups are selected and deselected.** To deselect a group, add a line modeled after those above. To select a group that is currently deselected, either delete its deselection line from the profile, or change the selected value from 0 to 1.
- **Changing which rules are selected and deselected.** To deselect a rule, change its selected value from 1 to 0; to select it, change its selected value from 0 to 1. In the example above, if an organization had a more stringent minimum password length requirement than the Enterprise Moderate profile specifies, then the checklist user could deselect the MinimumPasswordLength-8 rule and select the MinimumPasswordLength-12 rule.
- **Adding new groups or rules.** Checklist users may need to add new groups or rules; for example, if an organization has a minimum password length requirement of 10, and there is no existing rule that can check for that specific requirement, then a new rule would need to be created. Once the rule was created, the checklist user could then add a line to the profile, based on the existing lines, that referenced the rule’s ID and set it to be selected or deselected.

4.1.2 XCCDF Groups

An XCCDF group is a collection of rules and/or other groups. Groups can be defined in one of two ways: the group definition can include the definitions for all the rules and/or groups that belong to it, or the rules and/or groups that belong to the group can explicitly state in their individual definitions to which group they belong. The latter is a more modular approach that allows all rules to be defined within a single area of an XCCDF document. Figure 4 shows an example that uses both approaches. It is the definition of the group named IdentificationAndAuthentication, which is composed of seven other groups. Those seven groups are defined within the IdentificationAndAuthentication definition. However, notice that each of the additional group definitions is empty—no rules or other groups are listed.

```
<cdf:Group id="IdentificationAndAuthentication" weight="2">
   <cdf:title>Applicable 800-53 Identification and Authentication</cdf:title>
   <cdf:Group id="IA-1">
      <cdf:title>Identification and Authentication Policy and
         Procedures</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-2">
      <cdf:title>User Identification and Authentication</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-3">
      <cdf:title>Device Identification and Authentication</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-4">
      <cdf:title>Identifier Management</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-5">
      <cdf:title>Authenticator Management</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-6">
      <cdf:title>Authenticator Feedback</cdf:title>
   </cdf:Group>
   <cdf:Group id="IA-7">
      <cdf:title>Cryptographic Module Authentication</cdf:title>
   </cdf:Group>
</cdf:Group>
```

Figure 4: Example of XCCDF Group Definition

Figure 5 shows an example of a rule, MinimumPasswordLength-8, that includes a group membership declaration. The line `<requires idref="IA-5" />` indicates that the rule is part of the IA-5 group. Alternately, group membership could have been indicated by defining the MinimumPasswordLength-8 rule as part of the IA-5 group definition.

Groups are used for both management purposes (e.g., selecting or deselecting multiple rules at once) and for scoring purposes (e.g., using the default scoring model, scores are reported by group), so there are many reasons why they might need to be customized. Still, most customization involves adding groups or rules to a group, or removing groups or rules from a group. This can be done in two ways. One is to add group or rule definitions to the group definition or remove existing definitions from it. The other is to alter individual group or rule definitions to declare which groups and rules are part of the group; this is done by adding, removing, or altering the “requires idref” lines in the group and rule definitions.

Another reason to customize a group is to alter its weighting, which (if not the default) is specified in the first line of the group definition. Changing the weighing simply involves altering the weight number.

4.1.3 XCCDF Rules

An XCCDF rule is a high-level definition of a technical check on a system. Figure 5 shows an example of the code for a rule definition. The following briefly explains what the example contains:

- The first line assigns a rule ID and a weight.
- The second line gives a title for the rule.
- The third line (“description”) describes the intent of the rule.
- The fourth line (“reference”) provides a reference for more information about the requirement to which this rule maps.
- The fifth line (“requires”) specifies the group to which this rule belongs. In this case, the rule belongs to the IA-5 group.
- The next few lines (“check”) specify which low-level technical check should be performed (in this case, “oval:gov.nist.1:def:20”) and what XML file and XML should be used to perform that check.
- The last line indicates the end of the rule definition.

```
<Rule id="MinimumPasswordLength-8" weight="4">
   <title>Minimum Password Length = 8</title>
   <description>This setting specifies the minimum length of a password in
      characters. The rationale behind this setting is that longer passwords
      are more difficult to guess and crack than shorter passwords. The
      downside is that longer passwords are often more difficult for users to
      remember. Organizations that want to set a relatively large minimum
      password length should encourage their users to use passphrases, which
      may be easier to remember than conventional passwords.</description>
   <reference href="http://csrc.nist.gov/publications/nistpubs/800-53/SP800-
      53.pdf">IA-5</reference>
   <requires idref="IA-5" />
   <check system="http://oval.mitre.org/OVAL/XMLSchema/oval">
      <check-content-ref href="WindowsXP-SP800-68.xml"
         name="oval:gov.nist.1:def:20" />
   </check>
</Rule>
```

Figure 5: Example of XCCDF Rule Definition

This rule example could be customized in several ways, including the following:

- **Changing which technical checks the rule uses.** If the desired check is in the same XML file, simply alter the check's name in the "name" field. If the desired check is in a different XML file, update the check name, filename, and/or schema fields as needed, as shown in the "check" fields in the example.
- **Adding or removing technical checks.** A rule can perform more than one check; each check has its own "check" definition, like the one shown in Figure 5. So, adding a check simply involves adding another "check" definition, and removing a check is done by deleting the unwanted "check" definition.
- **Changing its weight.** This can be done by altering the weight number in the first line of the definition. However, profiles that use the rule can override the weight by specifying their own values, so altering the value in the definition might not necessarily cause the actual weighing to change. In such a case, it would be necessary to instead alter the profile definitions to use the desired weight for the rule.
- **Changing its group membership.** This is explained in Section 4.1.2.
- **Updating the reference.** If the reference needs updated (e.g., mapping the rule to a new requirement), simply edit the "reference" line.

4.1.4 XCCDF Values

An XCCDF value is a user-definable value that is referenced by a rule. Figure 6 shows an example of a definition for a value, AccountLockoutDurationTime. Users that need to change the value have two options. They can adjust what is assigned in the value statement (in this case, 15), which will affect all rules that use this value. Alternatively, they can leave the value code unchanged and instead alter the profiles that use this rule. For example, suppose that the AccountLockoutDurationTime value is referenced by eight profiles within an XCCDF document. To comply with the different policy requirements of each profile, the user might set the number in the value definition to the organization's standard value, and then alter the references to the value in the profiles that need to set it to something other than the standard value. The next-to-last line of Figure 3 shows an example of how such a value is set in a profile.

```
<cdf:value id="AccountLockoutDurationTime" type="number" operator="greater
than or equal">

   <cdf:title>Account Lockout Duration Time</cdf:title>

   <cdf:description> This value specifies how long the user account should be
      locked out. This is often set to a low but substantial value (e.g., 15
      minutes), for two reasons. First, a legitimate user that is
      accidentally locked out only has to wait 15 minutes to regain access,
      instead of asking an administrator to unlock the account. Second, an
      attacker who is guessing passwords using brute force methods will only
      be able to try a small number of passwords at a time, then wait 15
      minutes before trying any more. This greatly reduces the chances that
      the brute force attack will be successful.</cdf:description>

   <cdf:question>Account lockout duration time (in minutes)</cdf:question>

   <cdf:value>15</cdf:value>

   <cdf:default>15</cdf:default>

   <cdf:lower-bound>10</cdf:lower-bound>

   <cdf:upper-bound>30</cdf:upper-bound>

</value>
```

Figure 6: Example of XCCDF Value Definition

4.2 OVAL Customization

The official OVAL Web site includes an OVAL definition repository, where anyone can download accepted (approved) definitions.[15] Anyone may create their own definitions and submit them to the repository for consideration and approval, which is conducted through a formal review process. Typically, individual users do not need to customize existing OVAL definitions; instead, it is much more likely that users will write their own definitions. This section will present an example of an OVAL definition and its components—tests, objects, states, and variables—as a model for creating simple OVAL definitions. The OVAL Web site (http://oval.mitre.org/) contains extensive information on creating more complex and powerful OVAL definitions.

4.2.1 OVAL Definitions

As mentioned in Section 3.2, there are four types of OVAL definitions: vulnerability, patch, inventory, and compliance. All of these types of definitions are widely used in checklists. Because their construction is rather similar, this section will only review an example of a compliance definition, which is shown in Figure 7. The following briefly explains what the example contains:

- The first line assigns a definition ID, provides a version number for the definition, and designates the definition as a compliance definition.
- The group of lines tagged with “metadata” is, as the name implies, metadata for the definition. This includes the definition title, target platform, references, description, and NIST SP 800-53 control mapping.
- The next group of lines, tagged with “criteria”, lists the conditions to be checked when evaluating the system. The first criterion calls another definition, oval:gov.nist.1:def:9, that checks which platform the target system is using. The second criterion calls a test, oval:gov.nist.1:tst:16, that checks the minimum password length on the target system. The definition of this test is shown in Figure 8.
- The last line indicates the end of the definition.

[15] The repository is located at http://oval.mitre.org/repository/index.html.

```
<definition id="oval:gov.nist.1:def:20" version="1" class="compliance">
   <metadata>
      <title>Minimum Password Length of 8 Characters</title>
      <affected family="windows">
         <platform>Microsoft Windows XP</platform>
         <product>Windows XP, SP2, 32 bit</product>
      </affected>
      <reference source="NIST SP800-68 Appendix A" ref_id="1.4b"
         ref_url="http://csrc.nist.gov/itsec/download_WinXP.html" />
      <reference source="DISA FSO Checklist" ref_id="5.4.1.3" />
      <reference source="DISA VMS 6XID" ref_id="V0001106" />
      <reference source="DISA PDI ID" ref_id="1740" />
      <description>Minimum password length is 8 characters</description>
      <NIST>
         <SP800-53>IA-5</SP800-53>
      </NIST>
      <Additional_Reference>NSA NT Guide: Chap 5, p. 30; NSA WIN2K Guide,
         Group Policy: Security Configuration Toolset: Chap. 3, p. 22; NSA XP
         Guide: Chap. 4, p. 21; DODD 8500.1 Para 4.18; DODI 8500.2 DCCS-2,
         DCSC-1; CJCSM 6510.01 App. A, Enclosure A, Para. 5.b
         (8)</Additional_Reference>
   </metadata>
   <criteria>
      <extend_definition definition_ref="oval:gov.nist.1:def:9"
         comment="Precondition 9: Windows family, Windows XP, SP2, 32 bit" />
      <criterion test_ref="oval:gov.nist.1:tst:16" comment="Minimum password
         length is 8 characters" />
   </criteria>
</definition>
```

Figure 7: Example of OVAL Definition

4.2.2 OVAL Tests, Objects, States, and Variables

Section 4.2.1 mentions that OVAL tests are called as criteria within compliance definitions. Figure 8 shows an example of an OVAL test. The first line specifies the test's name (in this case, oval:gov.nist.1:tst:16), version, and XML schema reference, along with providing a comment describing the purpose of the test. The second line specifies the object to be accessed, and the third line specifies the state to be checked for that object.

```
<passwordpolicy_test id="oval:gov.nist.1:tst:16" version="1" check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   comment="Minimum password length is 8 characters">
   <object object_ref="oval:gov.nist.1:obj:8" />
   <state state_ref="oval:gov.nist.1:ste:21" />
</passwordpolicy_test>
```

Figure 8: Example of OVAL Test

An OVAL object specifies what aspect of the system will be checked. Figure 9 shows examples of three OVAL object definitions. The second object definition (which starts with "passwordpolicy") is for the object referenced in Figure 8. This object is the target system's password policy. In this case, the state definition (which was called in the Figure 8 example along with the object definition) specifies which portion of the password policy is to be evaluated. This is a different approach from the first object definition example in Figure 9, which references a specific value in the Windows registry.

```
<registry_object id="oval:gov.nist.1:obj:7" version="1"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
   <hive>HKEY_LOCAL_MACHINE</hive>
   <key>SYSTEM\CurrentControlSet\Control\Session Manager\Environment</key>
   <name>PROCESSOR_ARCHITECTURE</name>
</registry_object>
<passwordpolicy_object id="oval:gov.nist.1:obj:8" version="1"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows" />
<lockoutpolicy_object id="oval:gov.nist.1:obj:9" version="1"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows" />
```

Figure 9: Examples of OVAL Objects

As mentioned previously, a state definition gives the details of how a check will be performed. Figure 10 contains two examples of state definitions; the first one corresponds to the definition example presented throughout this section. The first line assigns a state ID and version number, as well as an XML schema reference. The second line specifies that the minimum password length value for the target system should be tested to see if it is greater than or equal to the value assigned to the variable named oval:gov.nist.1:var:8. The definition for that variable is shown in Figure 11.

```
<passwordpolicy_state id="oval:gov.nist.1:ste:21" version="1"
xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

   <min_passwd_len operation="greater than or equal"
      var_ref="oval:gov.nist.1:var:8" />

</passwordpolicy_state>

<passwordpolicy_state id="oval:gov.nist.1:ste:22" version="1"
xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

   <password_complexity operation="equals">1</password_complexity>

</passwordpolicy_state>
```

Figure 10: Examples of OVAL States

Of all the OVAL components discussed in this section—compliance definitions, tests, objects, states, and variables—variables are the most likely to be customized by end users. Typically this is to take into account organization-specific policies or environment-specific conditions. The first line of the example in Figure 11 shows the variable ID, version, data type (in this case, integer), and an explanatory comment. The second line ("value") assigns the value to the variable. This shows a value of 8 for the minimum password length. Changing this number would alter the value used by all states that reference this particular variable.

```
<constant_variable id="oval:gov.nist.1:var:8" version="1" datatype="int"
comment="Minimum password length">

   <value>8</value>

   <!--

   Title and description for oval:gov.nist.1:def:20 (Minimum Password Length
   of 8 Characters) must be changed if value is changed

   -->

</constant_variable>
```

Figure 11: Example of OVAL Variable

Appendix A—Glossary

Selected terms used in the publication are defined below.

Automated Checklist: A checklist that is applied through one or more tools that automatically alter or verify settings based on the contents of the checklist.

Checklist: See “Security Checklist”.

Custom Environment: An environment that contains systems in which the functionality and degree of security do not fit the other environments, Enterprise and Small Office/Home Office.

Enterprise Environment: An environment typically composed of large organizational systems with defined, organized suites of hardware and software configurations, usually consisting of centrally-managed workstations and servers protected from the Internet by firewalls and other network security devices.

Extensible Configuration Checklist Description Format (XCCDF): A language used to define structured collections of security configuration rules for sets of target systems.

Legacy Environment: A custom environment that contains older systems or applications that may use older, less-secure communication mechanisms.

Non-Automated Checklist: A checklist that is designed to be implemented manually, such as English prose instructions that describe the steps that an administrator should perform to secure a system or verify its security settings.

Open Vulnerability and Assessment Language (OVAL): A language used by security experts to exchange technical details about how to check for the presence of vulnerabilities and configuration issues on computer systems.

OVAL Definition: A set of OVAL instructions used to check a system for a security problem, such as an incorrect minimum password length setting.

OVAL Definition File: A file containing one or more OVAL definitions.

OVAL Object: A logical construct for a portion of a target system, such as a password policy, file, or Windows registry key.

OVAL State: A particular check of a specified OVAL object, such as verifying the existence of a file or the minimum password length required by a password policy.

OVAL Test: A specific system check that can act as a criterion for an OVAL definition.

Security Checklist: A document that contains instructions or procedures for securely configuring an IT product to an operational environment or verifying that an IT product has already been securely configured.

Security Content Automation: The process of using tools, scripts, and other technologies to automate the application or verification of security-related configuration settings for operating systems and applications.

Small Office/Home Office Environment: An environment that encompasses small, informal computer installations that are used for home or business purposes.

Specialized Security-Limited Functionality: A custom environment that contains systems and networks at high risk of attack or data exposure, with security taking precedence over functionality.

XCCDF Group: A logical construct in XCCDF that encompasses one or more XCCDF rules and/or one or more other XCCDF groups.

XCCDF Profile: A policy in XCCDF format that specifies which rules should be used to check a particular type of system.

XCCDF Rule: A high-level definition in XCCDF format of a technical check on a system. An XCCDF rule does not directly specify how a check should be performed, but instead points to other XML documents that contain the actual instructions for performing the check.

XCCDF Value: A user-definable value that can be used by one or more XCCDF rules.

Appendix B—Acronyms

Selected acronyms used in the publication are defined below.

C&A	Certification and Accreditation
CCE	Common Configuration Enumeration
CIS	Center for Internet Security
COTS	Commercial Off-the-Shelf
CVE	Common Vulnerabilities and Exposures
DHCP	Dynamic Host Configuration Protocol
DISA	Defense Information Systems Agency
DNS	Domain Name System
DOD	Department of Defense
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
FTP	File Transfer Protocol
GOTS	Government Off-the-Shelf
HIPAA	Health Information Portability and Accountability Act
IDS	Intrusion Detection System
IR	Interagency Report
IT	Information Technology
ITL	Information Technology Laboratory
MAC	Mission Assurance Category
NCP	National Checklist Program
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NVD	National Vulnerability Database
OMB	Office of Management and Budget
OVAL	Open Vulnerability and Assessment Language
SCAP	Security Content Automation Program
SMTP	Simple Mail Transfer Protocol
SOHO	Small Office/Home Office
SOX	Sarbanes-Oxley
SP	Special Publication
SSLF	Specialized Security-Limited Functionality
STIG	Security Technical Implementation Guide
URL	Uniform Resource Locator
US-CERT	United States Computer Emergency Readiness Team

VPN	Virtual Private Network
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language

Appendix C—Resources

The lists below provide examples of additional resources that may be helpful in better understanding the Security Content Automation Program and security checklists in general.

Checklist Resources

Resource	URL
Center for Internet Security (CIS)	http://cisecurity.org/
Common Vulnerabilities and Exposures (CVE)	http://cve.mitre.org/
Defense Information Systems Agency (DISA)	http://iase.disa.mil/stigs/checklist
FIPS PUB 199, *Standards for Security Categorization of Federal Information and Information Systems*	http://csrc.nist.gov/publications/fips/index.html
National Security Agency (NSA)	http://www.nsa.gov/snac/
National Vulnerability Database (NVD)	http://nvd.nist.gov/
NIST IR 7275, *Specification for the Extensible Configuration Checklist Description Format (XCCDF)*	http://checklists.nist.gov/docs/xccdf-spec-1.1.pdf
NIST Security Configuration Checklists Repository beta	http://checklists.nist.gov/
NIST SP 800-53, *Recommended Security Controls for Federal Information Systems*	http://csrc.nist.gov/publications/nistpubs/index.html
NIST SP 800-70, *Security Configuration Checklists Program for IT Products—Guidance for Checklists Users and Developers*	http://csrc.nist.gov/publications/nistpubs/index.html
Open Vulnerability and Assessment Language (OVAL)	http://oval.mitre.org/
XCCDF	http://checklists.nist.gov/xccdf.html