INTERNET-DRAFT
Intended Status: Informational
Expires: April 18, 2011

K. Landfield
McAfee

October 15, 2010

Naming Conventions for Vulnerabilities and Configurations
<draft-landfield-scap-naming-00.txt>

Status of this Memo

Copyright and License Notice

Abstract

This document provides an overview of the naming conventions currently in use for vulnerabilities and configurations.

Table of Contents

1 Introduction 3
1.1 Terminology 3
1.2 Purpose and Scope 3
1.3 Audience 3
1.4 Document Structure 3
2 Overview of Vulnerability Naming Schemes 3
2.1 Common Vulnerabilities and Exposures (CVE) 4
2.2 Common Configuration Enumeration (CCE) 5 5
3 Identifier Syntax 6
3.1 CVE Identifier Syntax 6
3.2 CCE Identifier Syntax 6
4 Security Considerations 6
4.1 Security Considerations for End User Organizations 6
4.1.1 Product and Service Selection and Design 7
4.1.2 Vulnerability Communications and Reporting 7
4.2 Security Considerations for Software Developer and Service Providers 8
4.2.1 Name Creation 8
4.2.2 Name Use 9
5 IANA Considerations 9
6 Security Considerations 9
7 IANA Considerations 9
8. Acknowledgements 10
9 References 10
9.1 Normative References 10
9.2 Informative References 10
Author’s Addresses 10

1 Introduction

This document provides an overview of the naming conventions currently in use for vulnerabilities and security configuration information.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2 Purpose and Scope

The purpose of this document is to provide recommendations for using vulnerability naming schemes. The document covers two schemes: CVE and CCE. The document gives an introduction to both schemes and makes recommendations for end-user organizations on using the names produced by these schemes. The document also presents recommendations for software and service vendors on how they should use vulnerability names and naming schemes in their product and service offerings.

1.3 Audience

The intended audience for this document is individuals who have responsibilities related to vulnerability management.

1.4 Document Structure

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

* Section 2 provides an overview of CVE and CCE.
* Section 3 describes the syntax of CVE and CCE identifiers.
* Section 4 gives recommendations to end-user organizations on using CVE and CCE, and makes recommendations for how IT product and service vendors should adopt CVE and CCE within their product and service offerings.
* Appendix A defines acronyms and abbreviations for the document.
* Appendix B lists related resources.

2 Overview of Vulnerability Naming Schemes

A vulnerability naming scheme is a systematic method for creating and maintaining a standardized dictionary of common names for a set of vulnerabilities in IT systems, such as software flaws in an operating system or security configuration issues in an application. The naming scheme ensures that each vulnerability entered into the dictionary

has a unique name. Using standardized vulnerability naming schemes supports interoperability. Organizations typically have many tools for system security management that reference vulnerabilities?for example, vulnerability and patch management software, vulnerability assessment tools, antivirus software, and intrusion detection systems. If these tools do not use standardized names for vulnerabilities, it may not be clear that multiple tools are referencing the same vulnerabilities in their reports, and it may take extra time and resources to resolve these discrepancies and correlate the information. This lack of interoperability can cause delays and inconsistencies in security assessment, reporting, decision-making, and vulnerability remediation, as well as hampering communications both within organizations and between organizations. Use of standardized names also helps minimize confusion regarding which problem is being addressed, such as which vulnerabilities a new patch mitigates. This helps organizations to quickly identify the information they need, such as remediation information, when a new problem arises. This publication provides information and recommendations related to the two most widely used vulnerability naming schemes: Common Vulnerabilities and Exposures (CVE), and Common Configuration Enumeration (CCE). Both are described in detail below.

2.1 Common Vulnerabilities and Exposures (CVE)

The CVE vulnerability naming scheme is for a dictionary of unique, common names for publicly known software flaws. CVE provides the following:

* A comprehensive list of publicly known software flaws
* A globally unique name to identify each vulnerability
* A basis for discussing priorities and risks of vulnerabilities
* A way for a user of disparate products and services to integrate vulnerability information

A CVE vulnerability entry consists of a unique identifier number, a short description of the vulnerability, and references to public advisories on the vulnerability. Figure 1 shows an example.

CVE ID: CVE-2000-0001

Description: RealMedia server allows remote attackers to cause a denial of service via a long ramgen request.

References:
BUGTRAQ: 19991222 RealMedia Server 5.0 Crasher (rmscrash.c)
BID:888

URL:http://www.securityfocus.com/bid/888

XF:realserver-ramgen-dos

Figure 1. Example CVE Entry

Working with researchers, The MITRE Corporation assigns CVE IDs to publicly known vulnerabilities in commercial and open source software. General information on CVE is available at http://cve.mitre.org/. The National Vulnerability Database (NVD), which is maintained by NIST, provides several ways of accessing CVE entries. NVD offers CVE search capabilities, as well as a variety of XML and RSS data feeds with CVE entries and supplemental information to support security automation technologies. NVD is publicly available at http://nvd.nist.gov/.

The MITRE Corporation maintains information on the use of CVE. Vendors that include CVE identifiers in their security advisories are listed at http://cve.mitre.org/compatible/alerts_announcements.html. Products and services that have been reviewed and evaluated by the MITRE Corporation and determined to be ?CVE-compatible?, which means that they meet a set of CVE requirements, are listed at http://cve.mitre.org/compatible/compatible.html.

2.2 Common Configuration Enumeration (CCE) 5

The CCE 5 vulnerability naming scheme is for a dictionary of names for software security configuration settings. Each type of security-related configuration issue is assigned a unique identifier to facilitate fast and accurate correlation of configuration data across multiple information sources and products.

There are five attributes in a CCE entry: a unique identifier number, a description of the configuration issue, logical parameters of the CCE, the associated technical mechanisms related to the CCE, and references to additional sources of information. Figure 2 provides an example of these attributes for a CCE 5 entry for Windows XP.

CCE ID: CCE-3108-8

Description: The correct service permissions for the Telnet service should be assigned.

Parameters: (1) set of accounts (2) list of permissions

Technical Mechanisms: (1) set via Security Templates (2) defined by Group Policy

References: Listed at http://cce.mitre.org/lists/cce_list.html

Figure 2. Example CCE Entry

The MITRE Corporation maintains and publishes the lists of CCE names. The lists, and additional information on CCE, are available at http://cce.mitre.org/.

3 Identifier Syntax

3.1 CVE Identifier Syntax

The CVE identifier is a string and is composed of three components spearated by hyphens: the CVE prefix, the year, and the vulnerability number.

The CVE prefix is the string "CVE".

The year is the year that the vulnerability was identified.

The vulnerability number is a four digit ASCII string and is assigned pseudo-randomly. This prevents guessing the next CVE identifier; people would attempt to guess the identifier and this would result in name collisions when they guessed incorrectly.

3.2 CCE Identifier Syntax

The CCE identifier is a string and is composed of three components separated by hyphens: the CCE prefix, a configuration identifier number, and a check digit.

The CCE prefix is the string "CCE".

The configuration identifier number is a variable length ASCII numeric string and is assigned pseudo-randomly. This prevents guessing the next CCE identifier; people would attempt to guess the identifier and this would result in name collisions when they guessed incorrectly.

The check value is a single numeric character. The Luhn check digit algorithm is used to generate the check value. [LUHN] The check digit addresses user entry problems, since administrators would routinely fat finger the identifier.

4 Security Considerations

4.1 Security Considerations for End User Organizations

This section provides recommendations for end-user organizations on how they should take advantage of the CVE and CCE specifications to improve their system security management. For all recommendations in this section involving an organization using CVE and CCE names, the organization should use the authoritative names; locations for downloading these are listed in Section 2. Some of the

recommendations in this section involve the Common Platform Enumeration (CPE) version 2.2 specification. CPE provides standardized, consistent names for referring to operating systems, hardware, and applications. CPE names are often used in conjunction with CVE and CCE names. Each CVE name and CCE name is related to one or more IT components, which can be expressed using CPE names. For example, a particular software flaw (identified by a CVE name) may affect seven products (identified by their CPE names). Using CPE names with CVE and CCE names further supports interoperability and standardization across products and services. The official CPE dictionary is available at http://nvd.nist.gov/cpe.cfm.

4.1.1 Product and Service Selection and Design

1. When evaluating IT products and services that use vulnerability names, such as for possible acquisition, organizations should take into consideration the products and services? support for CVE and/or CCE (as appropriate). Most organizations use a variety of products and services to detect, track, and mitigate vulnerabilities. Using standardized vulnerability names across products and services makes it much easier and faster to correlate information and to aggregate data from many disparate sources into a unified interface, such as a security dashboard.

2. Organizations developing their own custom security software that will use vulnerability names should ensure that the software uses authoritative CVE and/or CCE names, as appropriate, and uses CPE names with them when indicating which IT products the CVE and CCE names apply to. Using CVE, CCE, and CPE names supports interoperability with other software and services.

4.1.2 Vulnerability Communications and Reporting

1. Organizations should use authoritative CVE and CCE names in their internal vulnerability assessment and reporting, including assessment reports, notifications to system owners of detected vulnerabilities, and alerts identifying the vulnerabilities being targeted by active exploits. Use of CVE and CCE names will help to minimize confusion regarding which vulnerability is being referenced and whether a particular vulnerability has been mitigated. Organizations should also use CPE names when indicating which IT products the vulnerabilities apply to.

2. Organizations should use authoritative CVE and CCE names in their external vulnerability communications, as well as the CPE names for the IT products that the vulnerabilities apply to. For example, communications to incident response teams should reference, when known, the CVE or CCE names of the vulnerabilities that are being

exploited and the CPE names of the products that are being exploited. This ensures that incident communications precisely identify relevant vulnerabilities and affected products, enable correlation and integration of reports, and enable correlation with supplemental information residing in other data repositories. Another example is that organizations should use CVE and CCE names when communicating with vendors that support CVE and CCE names. Suppose that a vendor-supplied patch that purports to fix a vulnerability is defective; a statement to the vendor that a given CVE vulnerability remains after applying the patch conveys important information clearly and succinctly. Communications with vendors of scanning tools regarding false positives or false negatives will be clearer if the offending vulnerability is labeled by CVE or CCE name.

3. Organizations should encourage security software vendors to incorporate support for CVE and CCE into their products, as well as encourage all software vendors to include authoritative CVE and CCE names in their product security advisories and other security-related documentation and communications, as well as to reference the CPE names for their products.

4.2 Security Considerations for Software Developer and Service Providers

This section makes recommendations for how software developers and service providers should take advantage of CVE and CCE?s capabilities. This improves interoperability, thus increasing the efficiency of security management and improving the security of systems. Also, as described in the introduction to Section 3, using CPE names when indicating which products CVE and CCE names apply to further supports interoperability and standardization.

4.2.1 Name Creation

1. Software developers should participate in the CVE issuance processes to help ensure that CVEs provide the necessary information and are created in a timely manner, such as while planning a public vulnerability announcement or immediately after a vulnerability has been publicly announced. For more information on the CVE issuance process, see http://cve.mitre.org/cve/identifiers/index.html or contact cve@mitre.org.

2. Authors of CVE names should reference applicable vendor patch identification whenever possible.

3. Software developers are encouraged to work with the MITRE Corporation to establish unique CCE identifiers for their products? security configuration settings. Basic information on this is available at http://cce.mitre.org/lists/creation_process.html.

Software developers should contact the CCE Content Team at cce@mitre.org for more information on the process.

4.2.2 Name Use

1. Software developers and service providers should incorporate support for CVE and CCE into their products and services that reference publicly known software vulnerabilities. Also, CPE names should be used when indicating which products the CVE and CCE names apply to.

2. Software developers and service providers should incorporate the appropriate authoritative CVE and/or CCE names in their security advisories and other vulnerability-related documentation and communications, as well as the CPE names that indicate which products the CVE and/or CCE names apply to. Using CVE, CCE, and CPE names supports the use of standardized security automation technologies, which rely on common vulnerability and product names, and ensures clarity when referencing a given vulnerability.

5 IANA Considerations

This document has no actions for IANA.

6 Security Considerations

This document describes naming conventions and current processes for uniquely identify common security vulnerabilities and configuration information. These processes do not directly create new security vulnerabilities, and establishing a common language for discussing vulnerabilities and configuration information permits a more robust dialogue regarding security problems.

In some circumstances, disclosing configuration information may identify security vulnerabilities that could aid an attacker. When sharing information regarding specific systems, administrators may wish to employ encryption. However, administrators should recognize that an attacker may have other opportunities to identify the vulnerability, so encrypting this information is not a substitute for closing vulnerabilities!

7 IANA Considerations

This document has no actions for IANA.

Future standardization work may create registries for IANA to

maintain.

8. Acknowledgements

Much of the text in this document was taken from an ongoing revision of NIST Special Publication 800-51 by Dave Waltermire and Karen Scarfone. Tim Polk and Dave Waltermire helped get the document into Internet-Draft format.

This document was formatted using NroffEdit v 1.34 which is freely available from aaa-sec.com.

9 References

9.1 Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[LUHN] U.S. Patent 2,950,048, Computer for Verifying Numbers, Hans P. Luhn, August 23, 1960.

9.2 Informative References

[CCE] http://cce.mitre.org

[CVE] http://cve.mitre.org

Author’s Addresses

Kent Landfield
McAfee

EMail: kent_landfield@mcafee.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 21, 2011

T. Takahashi, Ed.
Y. Kadobayashi, Ed.
NICT

October 18, 2010

Cybersecurity Information Exchange Framework
draft-takahashi-cybex-intro-00

Abstract

The cybersecurity information exchange framework, known as CYBEX, is currently undergoing its first iteration of standardization efforts within ITU-T. The framework describes how cybersecurity information is exchanged between cybersecurity entities on a global scale and how the exchange is assured. This framework is intended to facilitate cybersecurity entities to work together beyond national and/or organizational boundaries.

Currently, ITU-T Draft Recommendation X.1500 defines the framework. The editors designated for the progress of the Draft Recommendation are (in alphabetical order): Inette Furey (DHS), Youki Kadobayashi (NICT), Bob Martin (MITRE), Angela Mckay (Microsoft), Stephen Adegbite (FIRST), Damir Rajnovic (FIRST), Gavin Reid (Cisco), Tony Rutkowski (Yaana), Gregg Schudel (Cisco).

On behalf of ITU-T Q.4/17, this draft introduces the overview of CYBEX in the IETF.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2011.

Table of Contents

1. Introduction 3
2. Conventions used in this document 3
3. Overview of Cybersecurity Information Exchange Framework . . . 4
4. IANA Considerations 5
5. Security Considerations 5
6. References 5
6.1. Normative References 5
6.2. Informative References 6
Appendix A. Additional Stuff 6
Authors' Addresses 6

1. Introduction

In the Internet, sources of threats cross borders of countries and even continents. Countermeasures against these cybersecurity threats, however, are most frequently implemented by individual organizations in isolation. Consequently, an organization in one country may be attacked by malware whose countermeasures are already known and implemented within another organization in another country. Such incidents occur due to the lack of sharing of information among organizations. One of the biggest factors preventing organizations from sharing information with each other is the absence of a globally common format and framework for cybersecurity information exchange. Albeit some countries such as the United States possess domestic standards for approaching this problem, most other countries have no such standards. Another such factor is the absence of assured information exchange framework, without which no organization will exchange information.

To cope with this problem, ITU-T is now building an emerging standard - The Cybersecurity Information Exchange Framework (CYBEX). CYBEX provides a globally common format and framework for assured cybersecurity information exchange, which will eventually minimize the disparity of cybersecurity information availability on a global scale. Since cybersecurity information can be shared worldwide, no country or organization implementing CYBEX will be left behind in terms of its availability. Consequently, developing countries, which currently have fewer resources to put towards cybersecurity, can become equal partners with developed countries with appropriate investments. Therefore countermeasures will be implemented through global collaboration. The framework will also advance the development of automating cybersecurity information exchange. Most cybersecurity information exchange within organizations are not currently automated and depend largely on human intervention. Email, telephone calls and even faceto- face meetings are still the primary method for information exchange. The need for and reliance on human interaction consumes a great deal of time. By advancing automation of cybersecurity information exchange, the costs (e.g., personnel costs) within each organization will be significantly reduced and the operation will be more efficient. At the same time, human-operation-based mistakes such as miscommunication can be avoided; thus the quality of operations can be improved.

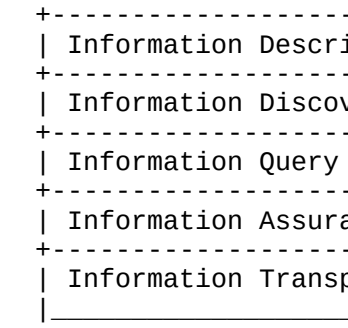
2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Overview of Cybersecurity Information Exchange Framework

CYBEX focuses on cybersecurity information exchange between cybersecurity organizations. Cybersecurity information is information required for cybersecurity operations such as on a vulnerability, and a cybersecurity organization is an organization running cybersecurity operations such as CSIRTs of countries and private companies. How to acquire/use cybersecurity information is outside the scope of CYBEX.

Considering the cybersecurity information life cycle, we observed that five functional blocks are needed for CYBEX: Information Description, Information Discovery, Information Query, Information Assurance and Information Transport, as are shown in Figure 1. The Information Description block structures cybersecurity information for exchange purposes, the Information Discovery block identifies and discovers cybersecurity information and entities, the Information Query block requests and responds with cybersecurity information, the Information Assurance block ensures the validity of the information, and Information Transport block exchanges cybersecurity information over networks.



Five functional blocks of CYBEX

Figure 1

Each functional block consists of assorted specifications as are shown in Table 1. As can be seen, one important characteristics of CYBEX is that this de jure standard is based on current de facto standards, and that by creating CYBEX in cooperation with the creators of each de facto standards we can increase the utility and compatibility of CYBEX with these standards, so users will be able to use CYBEX seamlessly with available products, making CYBEX more practical and deployable.

Functional blocks	CYBEX family specifications
Description	CPE, CCE, CVE, CWE, CAPEC, MAEC, CVSS, CWSS, OVAL, XCCDF, ARF, IODEF, CEE, TS102232, TS102667, TS23.271, RFC3924, EDRM, X.dexf, X.pfoc
Discovery	X.cybex.1, X.cybex-disc
Query	X.chirp
Assurance	EVCERT, TS102042 V2.0, X.eaa
Transport	TS102232-1, X.cybex-tp, X.cybex-beep

Table 1: CYBEX family specifications

Each of the functional blocks are elaborated on in the following subsections.

Further details are available at [ACMCCR_CYBEX].

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This paper introduced CYBEX, a new cybersecurity standard that will be finalized in December 2010. CYBEX provides a framework for assured cybersecurity information exchange between cybersecurity entities and minimizes the disparity of cybersecurity information availability among cybersecurity entities. The challenge is finding a means of permitting wide usage of CYBEX. Without global and widespread usage, CYBEX will not be able to provide its true value or contribute to cybersecurity. In order to advance cybersecurity, the effectiveness of CYBEX needs to be globally and widely recognized.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[ACMCCR_CYBEX]
Rutkowski, A., Kadobayashi, Y., Furey, I., Rajnovic, D., Martin, R., Takahashi, T., Schultz, C., Reid, G., Schudel, G., Hird, M., and S. Adegbite, "CYBEX - the Cybersecurity Information Exchange Framework (X.1500)", ACM SIGCOMM Computer Communication Review, Volume 40, Number 5, October 2010.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Takeshi Takahash (editor)
NICT
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Phone: +81 80 3490 2971
Email: takeshi_takahashi@nict.go.jp

Youki Kadobayashi (editor)
NICT
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: youki-k@is.aist-nara.ac.jp

SCAP Working Group
Internet Draft
Intended status: Informational
Expires: April 18, 2011

D. Waltermire
NIST

October 18, 2010

The Extensible Configuration Checklist Description Format (XCCDF) Version 1.1.4

draft-waltermire-scap-xccdf-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 18, 2009.

Copyright Notice

Abstract

This document specifies the data model and Extensible Markup Language (XML) representation for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.1.4. An XCCDF document is a structured collection of security configuration rules for some set 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 security guidance or checklist compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists and other configuration guidance, and thereby foster more widespread application of good security practices.

Table of Contents

1. Introduction.....5
2. Conventions used in this document.....6
3. Background.....6
   3.1. Motivation.....7
   3.2. Vision for Use.....8
   3.3. Summary of Changes since Version 1.0.....9
4. High-Level Requirements for XCCDF.....9
   4.1. Structure and Tailoring Requirements.....11
   4.2. Inheritance and Inclusion Requirements.....13
   4.3. Document and Report Formatting Requirements.....14
   4.4. Rule Checking Requirements.....14
   4.5. Test Results Requirements.....15
   4.6. Metadata and Security Requirements.....16
5. Data Model.....17
   5.1. Benchmark Structure.....19
      5.1.1. Inheritance.....20
   5.2. Object Content Details.....21
      5.2.1. Benchmark.....21
      5.2.2. Item.....24
         5.2.2.1. Group::Item.....26
         5.2.2.2. Rule::Item.....28
         5.2.2.3. Value::Item.....33
      5.2.3. Profile.....36
      5.2.4. TestResult.....38

5.2.4.1. TestResult/rule-result.....41
5.3. Processing Models.....44
5.3.1. Loading Processing Sequence.....45
5.3.2. Traversal Processing Sequence.....48
5.3.2.1. Benchmark Processing Algorithm.....48
5.3.2.2. Item Processing Algorithm.....49
5.3.3. Substitution Processing.....51
5.3.4. Rule Application and Compliance Scoring.....51
5.3.5. Scoring and Results Model.....52
5.3.6. Score Computation Algorithms.....54
5.3.6.1. The Default Model.....54
5.3.6.2. The Flat Model.....55
5.3.6.3. The Flat Unweighted Model.....56
5.3.6.4. The Absolute Model.....56
5.3.7. Multiply-Instantiated Rules.....56
6. XML Representation.....57
6.1. XML Document General Considerations.....57
6.2. XML Element Dictionary.....58
6.2.1. <Benchmark>.....59
6.2.2. <Group>.....59
6.2.3. <Rule>.....61
6.2.4. <Value>.....62
6.2.5. <Profile>.....63
6.2.6. <TestResult>.....64
6.2.7. Other Elements and Attributes.....65
6.2.7.1. <benchmark>.....66
6.2.7.2. <check>.....66
6.2.7.3. <check-import>.....67
6.2.7.4. <check-export>.....68
6.2.7.5. <check-content>.....68
6.2.7.6. <check-content-ref>.....68
6.2.7.7. <choices>.....69
6.2.7.8. <choice>.....69
6.2.7.9. <complex-check>.....70
6.2.7.10. <conflicts>.....71
6.2.7.11. <cpe-list>.....72
6.2.7.12. <default>.....72
6.2.7.13. <description>.....72
6.2.7.14. <fact>.....73
6.2.7.15. <fix>.....73
6.2.7.16. <fixtext>.....76
6.2.7.17. <front-matter>.....77
6.2.7.18. <ident>.....77
6.2.7.19. <identity>.....77
6.2.7.20. <impact-metric>.....78
6.2.7.21. <instance>.....79
6.2.7.22. <lower-bound>.....80

6.2.7.23. <match>.....80
6.2.7.24. <message>.....81
6.2.7.25. <metadata>.....81
6.2.7.26. <model>.....82
6.2.7.27. <new-result>.....83
6.2.7.28. <notice>.....83
6.2.7.29. <old-result>.....83
6.2.7.30. <organization>.....84
6.2.7.31. <override>.....84
6.2.7.32. <param>.....85
6.2.7.33. <plain-text>.....86
6.2.7.34. <platform>.....86
6.2.7.35. <platform-specification>.....87
6.2.7.36. <platform-definitions>, <Platform-Specification>
.....87
6.2.7.37. <profile>.....88
6.2.7.38. <profile-note>.....88
6.2.7.39. <question>.....88
6.2.7.40. <rationale>.....89
6.2.7.41. <rear-matter>.....89
6.2.7.42. <reference>.....89
6.2.7.43. <refine-rule>.....90
6.2.7.44. <refine-value>.....91
6.2.7.45. <remark>.....92
6.2.7.46. <requires>.....92
6.2.7.47. <result>.....93
6.2.7.48. <rule-result>.....93
6.2.7.49. <score>.....94
6.2.7.50. <select>.....94
6.2.7.51. <set-value>.....95
6.2.7.52. <signature>.....95
6.2.7.53. <source>.....96
6.2.7.54. <status>.....96
6.2.7.55. <sub>.....97
6.2.7.56. <target>.....97
6.2.7.57. <target-address>.....97
6.2.7.58. <target-facts>.....98
6.2.7.59. <title>.....98
6.2.7.60. <upper-bound>.....98
6.2.7.61. <value>.....99
6.2.7.62. <version>.....99
6.2.7.63. <warning>.....100
6.3. Handling Text and String Content.....100
6.3.1. XHTML Formatting and Locale.....100
6.3.2. String Substitution and Reference Processing.....101
7. Security Considerations.....103
8. IANA Considerations.....103

9. Conclusions.....103
10. References.....104
10.1. Normative References.....104
10.2. Informative References.....105
11. Acknowledgments.....105
Appendix A. Pre-Defined URIs.....106
A.1. Long-Term Identification Systems.....106
A.2. Check Systems.....106
A.3. Scoring Models.....107
A.4. Target Platform Facts.....107
A.5. Remediation Systems.....108

1. Introduction

This document describes version 1.1.4 of the Extensible Configuration Checklist Description Format (XCCDF), a standard data model and processing discipline for supporting secure configuration and assessment. The requirements and goals are explained in the main content; however, in summary, this document addresses:

- Document generation
- Expression of policy-aware configuration rules
- Support for conditionally applicable, complex, and compound rules
- Support for compliance report generation and scoring
- Support for customization and tailoring.

The model and its Extensible Markup Language (XML) representation are intended to be platform-independent and portable, to foster broad adoption and sharing of checklist rules. The processing discipline of the format requires, for some uses, a service layer that can collect and store system information and perform simple policy-neutral tests against the system information; this is true for technical and non-technical applications of XCCDF. These conditions are described in detail below.

The XML schema for XCCDF can be found at: http://scap.nist.gov/schema/xccdf/1.1/xccdf-1.1.4.xsd.

NOTE: This document is being published for the use of the internet community for evaluation of SCAP. While this document is not subject to change, change control would be granted to the IETF to support future security content automation standardization work. It

is expected that this document would be published as an RFC to document the starting point of the effort.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Background

This section provides informative background material on XCCDF, including the rationale behind its creation and the ways in which it can be used.

To promote the use, standardization, and sharing of effective security checklists, the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) have collaborated with representatives of private industry to develop the XCCDF specification. The specification is vendor-neutral, flexible, and suited for a wide variety of checklist applications. XCCDF was originally intended to be used for technical security checklists. Although this is still the primary use of XCCDF, XCCDF also has extensions into non-technical applications (e.g., owner’s manuals, user guides, non-technical security controls, and items considered "manual procedures").

The security of an information technology (IT) system typically can be improved if the identified software flaws and configuration settings that affect security are properly addressed. The security of an IT system may be measured in a variety of ways; one operationally relevant method is determining conformance of the system configuration to a specified security baseline, guidance document, or checklist. These typically include criteria and rules for hardening a system against the most common forms of compromise and exploitation, and for reducing the exposed "attack surface" of a system. Many companies, government agencies, community groups, and product vendors generate and disseminate security guidance documents. While definition of the conditions under which a security setting should be applied can differ among the guidance documents, the underlying specification, test, and report formats used to identify and remediate said settings tend to be specialized and unique.

Configuring a system to conform to specified security guidance or other security specifications is a highly technical task. To aid system administrators, commercial and community developers have created automated tools that can both determine a system's conformance to a specified guideline and provide or implement remediation measures. Many of these tools are data-driven in that they accept a security guidance specification in some program-readable form (e.g., XML, .inf, csv), and use it to perform the checks and tests necessary to measure conformance, generate reports, and perhaps remediate as directed. However, with rare exceptions, none of these tools (commercial or government developed) employ the same data formats. This unfortunate situation perpetuates a massive duplication of effort among security guidance providers and provides a barrier for content and report interoperability.

3.1. Motivation

Today, groups promoting good security practices and system owners wishing to adopt them face an increasingly large and complex problem. As the number of IT systems increases, automated tools are necessary for uniform application of security rules and visibility into system status. These conditions have created a need for mechanisms that:

- Ensure compliance to multiple policies (e.g., Payment Card Industry Data Security Standards (PCI-DSS), US Federal Information Security Management Act (FISMA), US Health Insurance Portability and Accountability Act (HIPAA))

- Permit faster, more cooperative, and more automated definition of security rules, procedures, guidance documents, alerts, advisories, and remediation measures

- Permit fast, uniform, manageable administration of security checks and audits

- Support the evaluation of the health of systems connecting to a network using NAC/NAP/TNC related protocols.

- Permit composition of security rules and tests from different community groups and vendors

- Permit scoring, reporting, and tracking of security status and checklist conformance, both over distributed systems and over the same systems across their operational lifetimes

- Foster development of interoperable community and commercial tools for creating and employing security guidance and checklist data.

Today, such mechanisms exist only in some isolated niche areas (e.g., Microsoft Windows patch validation) and they support only narrow slices of security guidance compliance functionality. For example, patch checking and secure configuration guidance often are not addressed at the same level of detail (or at all) in a single guidance document; however, both are required to secure a system against known attacks. This document defines a data model and format specification for an extensible, interoperable checklist language that is capable of including both technical and non-technical requirements in the same XML document.

3.2. Vision for Use

XCCDF is designed to enable easier, more uniform creation of security checklists and procedural documents, and allow them to be used with a variety of commercial, Government off-the-shelf (GOTS), and open source tools. The motivation for this is improvement of security for IT systems, including the Internet, by better application of known security practices and configuration settings.

One potential use for XCCDF is streamlining compliance to security requirements. Government agencies and the private sector have difficulty measuring the security of their IT systems. They also struggle to implement technical policy, such as laws and regulations, and then to demonstrate unambiguously to various audiences (e.g., Inspector General, auditor) that they have complied and ultimately improved the security of their systems. This difficulty arises from various causes, such as different interpretations of policy, system complexity, and human error. XCCDF proposes to automate certain technical aspects of security by converting human-readable text contained in various publications (e.g., configuration guides, checklists, vulnerability databases) into a machine-readable XML format such that the various audiences (e.g., scanning vendors, checklist/configuration guide, auditors) will be operating in the same semantic context. The end result will allow organizations to use commercial off-the-shelf (COTS) tools to automatically check their security and map to technical compliance requirements.

The scenarios below illustrate some uses of security checklists and tools that XCCDF will foster.

- Scenario 1 - An academic group produces a checklist for secure configuration of a particular server operating system version. A government organization issues a set of rules extending the academic checklist to meet more stringent user authorization criteria imposed by statute. A medical enterprise downloads both the academic checklist and the government extension, tailors the combination to fit their internal security policy, and applies an enterprise-wide audit using a commercial security audit tool. Reports output by the tool include remediation measures which the medical enterprise IT staff can use to bring their systems into full internal policy compliance.

- Scenario 2 - A government-funded lab issues a security advisory about a new Internet worm. In addition to a prose description of the worm's attack vector, the advisory includes a set of short checklists in a standard format that assess vulnerability to the worm for various operating system platforms. Organizations all over the world pick up the advisory, and use installed tools that support the standard format to check their status and fix vulnerable systems.

- Scenario 3 - An industry consortium, in conjunction with a product vendor, wants to produce a security checklist for a popular commercial server. The core security settings are the same for all operating system (OS) platforms on which the server runs, but a few settings are OS-specific. The consortium crafts one checklist in a standard format for the core settings, and then writes several OS-specific ones that incorporate the core settings by reference. Users download the core checklist and the OS-specific extensions that apply to their installations, and then run a checking tool to score their compliance with the checklist.

4. High-Level Requirements for XCCDF

The general objective for XCCDF is to allow security analysts and IT experts to create effective and interoperable automated checklists, and to support the use of automated checklists with a wide variety of tools.

The following items describe the most common high-level workflow for XCCDF creation and use, and list corresponding requirements for the design of XCCDF:

1. Step 1: Security and domain experts create a security guidance document or checklist, which is an organized collection of rules about a particular kind of system or platform. To support this use, XCCDF must be an open, standardized format, amenable to generation by and editing with a variety of tools. It must be expressive enough to represent complex conditions and relationships about the systems to be assessed, and it must also incorporate descriptive material and remediative measures. (XCCDF Benchmarks may include specification of the hardware and/or software platforms to which they apply; however, it is recommended that programmatically ascertainable information should be relegated to the lower-level identification and remediation languages. For specifying programmatically ascertainable information in the XCCDF file, the specification should be concrete and granular so that compliance checking tools can detect if a Rule is suited for a target platform.)

2. Step 2: Auditors and system administrators may employ tailoring tools to customize a security guidance document or checklist for their local environment or policies. For example, certain organizations may have trouble applying all settings without exception. For those settings which hinder functionality (perhaps with legacy systems, or in a hybrid Windows 2000/2003 domain), an organization may wish to tailor the corresponding XML. For this reason, an XCCDF document must include the structure and interrogative text required to direct the user through the tailoring of a Benchmark, and it must be able to hold or incorporate the user’s tailoring responses. For example, a checklist user might want to set the password policy requirement to be more or less stringent than the provided security recommendations. XCCDF should be extensible to allow for the custom tailoring and inclusion of the explanatory text for deviation from recommended policy.

3. Step 3: Tools process the XCCDF document and generate output. The primary use case for an XCCDF-formatted security guidance document is to facilitate the normalization of configuration content through automated security tools. Such tools should accept one or more XCCDF documents along with supporting system test definitions, and determine whether or not the specified rules are satisfied by a target system. The XCCDF document should support generation of a compliance report, including a weighted compliance score. Additional XCCDF design requirements related to document processing and output are as follows:

- In addition to a report, some tools may utilize XCCDF-formatted content (and associated content from other tools) to bring a system into compliance through the remediation of identified vulnerabilities or misconfigurations. XCCDF must be able to encapsulate the remediation scripts or texts, including several alternatives.

- XCCDF documents might also be used in vulnerability scanners, to test whether or not a target system is vulnerable to a particular kind of attack. For this purpose, the XCCDF document would play the role of a vulnerability alert, but with the ability to both describe the problem and drive the automated verification of its presence.

- Although the goal of XCCDF is to distill human-readable prose checklists into machine-readable XML, XCCDF should be structured to foster the generation of human-readable prose documents from XCCDF-format documents.

- The structure of an XCCDF document should support transformation into HTML, for posting the security guidance as a Web page.

- An XCCDF document should be transformable into other XML formats, to promote portability and interoperability.

In addition to these design requirements, an XCCDF document should be amenable to embedding inside other documents. Likewise, XCCDF's extensibility should include provisions for incorporating other data formats. And finally, XCCDF must be extensible to include new functionality, features, and data stores without hindering the functionality of existing XCCDF-capable tools.

4.1. Structure and Tailoring Requirements

The basic unit of structure for a security guidance document or checklist is a rule. A rule simply describes a state or condition which the target of the document should exhibit. A simple document might consist of a simple list of rules, but richer ones require additional structure.

To support customization of the standardized XML format and subsequent generation of documents by and for consumers, XCCDF must allow authors to impose organization within the document. One such basic requirement is that authors will need to put related rules into named groups.

An author must be able to designate the order in which rules or groups are processed. In the simplest case, rules and groups are processed in sequence according to their location in the XCCDF document.

To facilitate customization, the author SHOULD include descriptive and interrogative text to help a user make tailoring decisions. The following two customization options are available:

- Selectability - A tailoring action might select or deselect a rule or group of rules for inclusion or exclusion from the security guidance document. For example, an entire group of rules that relate to physical security might not apply if one were conducting a network scan. In this case, the group of rules delineated under the physical security group could be deselected. For example, certain rules might apply according to the impact rating of the system. For this purpose, systems that have an impact rating of Low might not have all of the same access control requirements as a system with a High impact rating, and therefore the those rules that are not applicable for the Low system can be deselected.

- Substitution - A tailoring action might substitute a locally-significant value for a general value in a rule. For example, at a site where all logs are sent to a designated logging host, the address of that log server might be substituted into a rule about audit configuration. Another example is a system with an impact rating of High that requires a 12-character password, and a system with an impact rating of Moderate that only requires an 8-character password. Depending on the impact rating of the target system, the user can customize or tailor the value through substitution.

When customizing security guidance documents, the possibility arises that some rules within the same document might conflict or be mutually exclusive. To avert potential problems, the author of a security guidance document must be able to identify particular tailoring choices as incompatible, so that tailoring tools can take appropriate actions.

In addition to specifying rules, XCCDF must support structures that foster use and re-use of rules. To this end, XCCDF must provide a means for related rules to be grouped and for sets of rules and groups to be designated, named, and easily applied. Examples of this requirement are demonstrated by the US Defense Information Systems Agency (DISA) Gold and Platinum levels (Gold being the less stringent of the two levels). NIST also provides distinctions

according to environment and impact rating (High, Moderate, or Low) [10]. Likewise, the Center for Internet Security (CIS) designates multiple numeric levels for their checklists (e.g., Level 1, Level 2).

To facilitate XCCDF adoption for the aforementioned requirements, XCCDF provides two basic processing modes: rule checking and document generation. It must be possible for a security guidance or checklist author to designate the modes (e.g., Gold, Platinum, High Impact Rating, Level 1) under which a rule should be processed.

4.2. Inheritance and Inclusion Requirements

Some use cases require XCCDF to support mechanisms for authors to extend (inherit from) existing rules and rule groups, in addition to expressing rules and groups in their entirety. For example, it must be possible for one XCCDF document to include all or part of another as demonstrated in the following scenarios:

- An organization might choose to define a foundational XCCDF document for a family of platforms (e.g., Unix-like operating systems) and then extend it for specific members of the family (e.g., Solaris) or for specific roles (e.g., mail server).

- An analyst might choose to make an extended version of an XCCDF document by adding new rules and adjusting others.

- If the sets of rules that constitute an XCCDF document come from several sources, it is useful to aggregate them using an inclusion mechanism. (Note: The XCCDF specification does not define its own mechanisms for inclusion; instead, implementations of XCCDF tools should support the XML Inclusion (XInclude) facility standardized by the World Wide Web Consortium (W3C) [2].)

- Within an XCCDF document, it is desirable to share descriptive material among several rules, and to allow a specialized rule to be created by extending a base rule.

- For updating an XCCDF document, it is convenient to incorporate changes or additions using extensions.

- To allow broader site-specific or enterprise-specific customization, a user might wish to override or amend any portion of an XCCDF rule.

4.3. Document and Report Formatting Requirements

Generating human-readable prose documents from the underlying XCCDF constitutes a primary use case. Authors require mechanisms for formatting text, including images, and referencing other information resources. These mechanisms must be separable from the text itself, so each can be filtered out by applications that do not support or require them. (XCCDF 1.1.4 currently satisfies these formatting requirements mainly by allowing inclusion of XHTML markup tags [3].)

The XCCDF language must also allow for the inclusion of content that does not contribute directly to the technical content. For example, authors tend to include 'front matter' such as an introduction, a rationale, warnings, and references. XCCDF allows for the inclusion of intra-document and external references and links.

4.4. Rule Checking Requirements

One of XCCDF's main features is the organization and selection of target-applicable groups and rules for performing security and operational checks on systems. Therefore, XCCDF must have access to granular and expressive mechanisms for checking the state of a system according to the rule criteria. The model for this requirement includes the notion of collecting or acquiring the state of a target system, and then checking the state for conformance to conditions and criteria expressed as rules. The operations used have varied with different existing applications; some rule checking systems use a database query operation model, others use a pattern-matching model, and still others load and store state in memory during execution. Rule checking mechanisms used for XCCDF must satisfy the following criteria:

- The mechanism must be able to express both positive and negative criteria. A positive criterion means that if certain conditions are met, then the system satisfies the check, while a negative criterion means that if the conditions are met, the system fails the check. Experience has shown that both kinds are necessary when crafting criteria for checks.

- The mechanism must be able to express boolean combinations of criteria. It is often impossible to express a high-level security property as a single quantitative or qualitative statement about a system's state. Therefore, the ability to combine statements with 'and' and 'or' is critical.

- The mechanism must be able to incorporate tailoring values set by the user. As described above, substitution is important for XCCDF document tailoring. Any XCCDF checking mechanism must support substitution of tailored values into its criteria or statements as well as tailoring of the selected set of rules.

A single rule specification scheme (e.g., Open Vulnerability and Assessment Language (OVAL) [11]) may not satisfy all potential uses of XCCDF. To facilitate other lower-level rule checking systems, XCCDF supports referencing by including the appropriate file and check reference in the XCCDF document. It is important that the rule checking system be defined separately from XCCDF itself, so that both XCCDF and the rule checking system can evolve and be used independently. This duality implies the need for a clear interface definition between XCCDF and the rule checking system, including the specification of how information should pass from XCCDF to the checking system and vice versa.

4.5. Test Results Requirements

Another objective of XCCDF is to facilitate a standardized reporting format for automated tools. In many organizations, several COTS and GOTS products are used to determine the security of IT systems and their compliance to various stated polices. Unfortunately, the outputs from these tools are not standardized and therefore costly customization can be required for trending, aggregation, and reporting. Addressing this use case, XCCDF provides a means for storing the results of the rule checking subsystem.

Security tools sometimes include only the results of the test or tests in the form of a pass/fail status. Other tools provide additional information so that the user does not have to access the system to determine additional information (e.g., instead of simply indicating that more than one privileged account exists on a system, certain tools also provide the list of privileged accounts). Independent of the robust or minimal reporting of the checking subsystem, the following information is basic to all results:

- The security guidance document or checklist used, along with any adaptations via customization or tailoring applied

- Information about the target system to which the test was applied, including arbitrary identification and configuration information about the target system

- The time interval of the test, and the time instant at which each individual rule was evaluated

- One or more compliance scores
- References to lower-level details possibly stored in other output files.

4.6. Metadata and Security Requirements

As the recognized need for security increases, so does the number of recommended security configuration guidance documents. The DISA Security Technical Implementation Guides (STIGs) and accompanying checklist documents have been available under http://iase.disa.mil/stigs for many years. Likewise, NIST's interactions with vendors and agencies have yielded checklist content provided at http://checklists.nist.gov/. NSA maintains a web site offering security guidance at http://www.nsa.gov/snac/, and CIS provides checklist content at http://cisecurity.org/. Likewise, product vendors such as Microsoft Corporation, Sun Microsystems, Apple Computer, and Hewlett-Packard (to name a few) are providing their own security guidance documents independent of traditional user guides.

The majority of these checklists exist in various repositories in English prose format; however, there is a recognized need and subsequent migration effort to represent said checklists in standardized XML format. To facilitate discovery and retrieval of security guidance documents in repositories and on the open Internet, XCCDF must support inclusion of metadata about a document. Some of the metadata that must be supported include: title, name of author(s), organization providing the guidance, version number, release date, update URL, and a description. Since a number of metadata standards already exist, it is preferable that XCCDF simply incorporate one or more of them rather than defining its own metadata model.

In addition to specifying rules to which a target system should comply, an XCCDF document must support mechanisms for describing the steps to bring the target into compliance. While checking compliance to a given security baseline document is common, remediation of an IT system to the recommended security baseline document should be a carefully planned and implemented process. Security guidance users should be able to trust security guidance documents, especially if they intend to accept remediation advice from them. Therefore, XCCDF must support a mechanism whereby guidance users can validate the integrity, origin, and authenticity of guidance documents.

Digital signatures are the natural mechanism to satisfy these integrity and proof-of-origin requirements. Fortunately, mature

standards for digital signatures already exist that are suitable for asserting the authorship and protecting the integrity of guidance documents. XCCDF must provide a means to hold such signatures, and a uniform method for applying and validating them.

5. Data Model

The fundamental data model for XCCDF consists of four main object data types:

1. Benchmark. An XCCDF document SHALL hold exactly one Benchmark object. A Benchmark SHOULD hold descriptive text, and SHOULD act as a container for Items and other objects.

2. Item. An Item is a named constituent of a Benchmark; it MAY hold descriptive text, and SHALL be referenced by a unique id. There are several derived classes of Items:

- Group. This kind of Item SHOULD hold other Items. A Group SHALL be either selected or unselected. (If a Group is unselected, then all of the Items it contains SHALL be implicitly unselected.)

- Rule. This kind of Item MAY hold check references and a scoring weight, and MAY also hold remediation information. A Rule SHALL be either selected or unselected.

- Value. This kind of Item is a named data value that MAY be substituted into other Items' properties or into checks. It MAY have an associated data type and metadata that express how the value should be used and how it can be tailored.

3. Profile. A Profile is a collection of attributed references to Rule, Group, and Value objects. It supports the requirement to allow definition of named levels or baselines in a Benchmark (see Section 4.1).

4. TestResult. A TestResult object SHALL hold the results of performing a compliance test against a single target device or system.

Figure 1 shows the data model relationships as a Unified Modeling Language (UML) diagram. As shown in the figure, one Benchmark MAY hold many Items, but each Item SHALL belong to exactly one Benchmark. Similarly, a Group MAY hold many Items, but an Item SHALL belong to only one Group. Thus, the Items in an XCCDF document form a tree, where the root node is the Benchmark, interior nodes are Groups, and the leaves are Values and Rules.

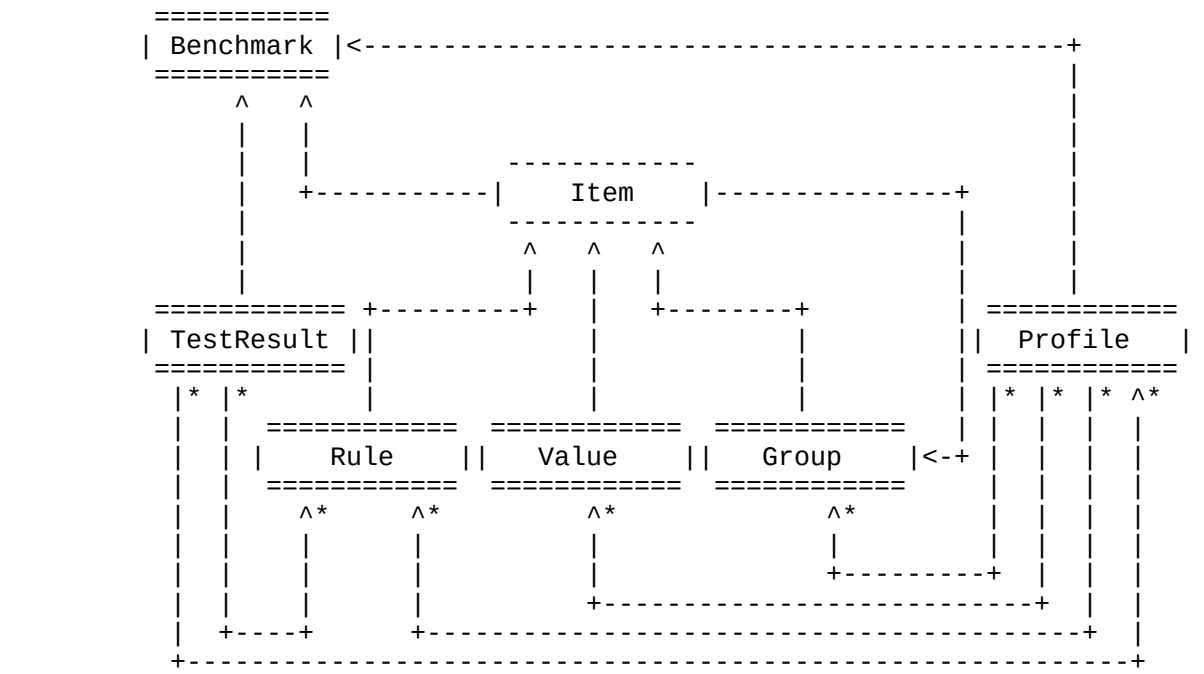


Figure 1 - XCCDF High-Level Data Model

A Profile object MAY reference Rule, Value, and Group objects. A TestResult object SHALL reference Rule objects and MAY also reference a Profile object.

The definition of a Value, Rule, or Group MAY extend another Value, Rule, or Group. The extending Item SHALL inherit property values from the extended Item. This extension mechanism is separate and independent of grouping.

Group and Rule items MAY be marked by a Benchmark author as selected or unselected. A Group or Rule that is not selected SHALL NOT undergo processing. The author MAY also stipulate, for a Group, Rule, or Value, whether or not the end user is permitted to tailor it.

Rule items MAY have a scoring weight associated with them, which can be used by a Benchmark checking tool to compute a target system’s overall compliance score. Rule items MAY also hold remediation information.

Value items MAY include information about current, default, and permissible values for the Value. Each of these properties of a Value MAY have an associated selector id, which is used when customizing the Value as part of a Profile. For example, a Value might be used to hold a Benchmark's lower limit for password length on some operating system. In a Profile for that operating system to be used in a closed lab, the default value might be 8, but in a Profile for that operating system to be used on the Internet, the default value might be 12.

5.1. Benchmark Structure

Typically, a Benchmark SHOULD hold one or more Groups, and each group SHOULD hold some Rules, Values, and additional child Groups. Figure 2 illustrates this relationship, and the order in which the contents of a Benchmark MUST appear.

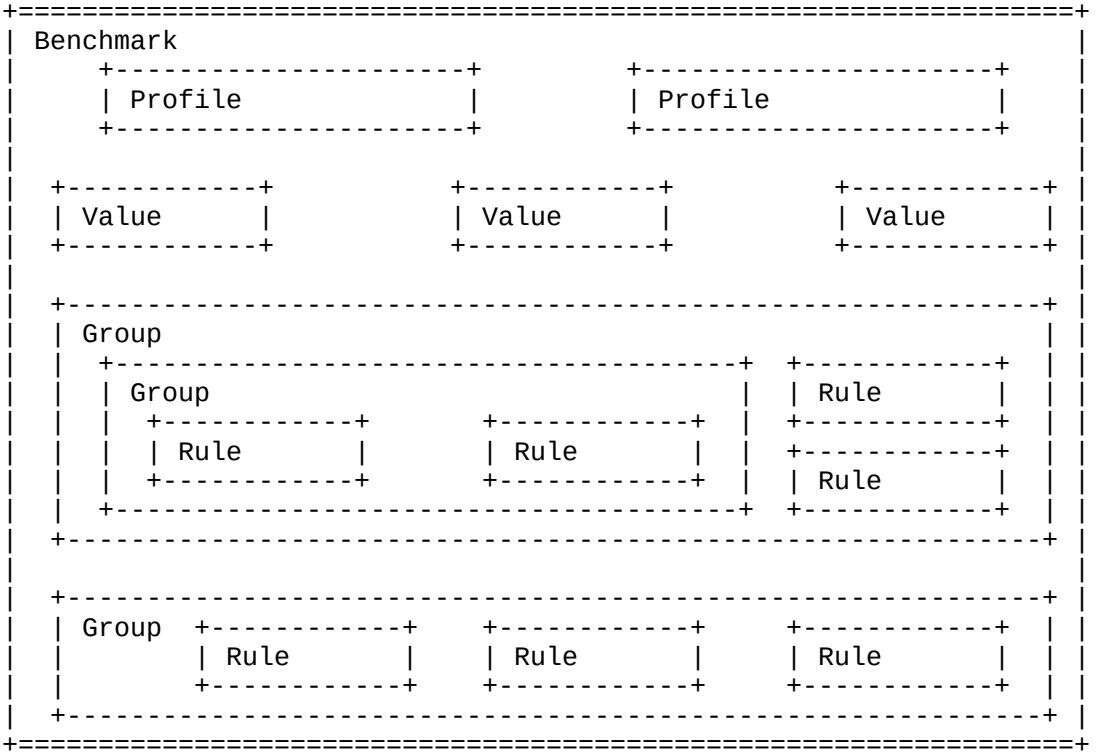


Figure 2 - Typical Structure of a Benchmark

Groups allow a Benchmark author to collect related Rules and Values into a common structure and provide descriptive text and references about them. Further, groups allow Benchmark users to select and deselect related Rules together, helping to ensure commonality among users of the same Benchmark. Lastly, groups affect Benchmark compliance scoring. As Section 5.3 explains, an XCCDF compliance score SHALL be calculated for each group, based on the Rules and Groups in it. The overall XCCDF score for the Benchmark SHALL be computed only from the scores on the immediate Group and Rule children of the Benchmark object. In the tiny Benchmark shown in Figure 2, the Benchmark score would be computed from the scores of Group (d) and Group (j). The score for Group (j) would be computed from Rule (l) and Rule (m).

5.1.1. Inheritance

The possible inheritance relations between Item object instances are constrained by the tree structure of the Benchmark, but are otherwise independent of it. In other words, all extension relationships MUST be resolved before the Benchmark can be used for compliance testing. An Item SHALL only extend another Item of the same type that is ’visible’ from its scope. In other words, an Item Y MAY extend a base Item X, as long as they are the same type, and one of the following visibility conditions holds:

1. X is a direct child of the Benchmark.

2. X is a direct child of a Group which is also an ancestor of Y.

3. X is a direct child of a Group which is extended by any ancestor of Y.

For example, in the tiny Benchmark structure shown in Figure 2, it would be legal for Rule (g) to extend Rule (f) or extend Rule (h). It would not be legal for Rule (i) to extend Rule (m), because (m) is not visible from the scope of (i). It would not be legal for Rule (l) to extend Group (g), because they are not of the same type.

The ability for a Rule or Group to be extended by another gives Benchmark authors the ability to create variations or specialized versions of Items without making copies.

5.2. Object Content Details

The lists below show the properties that make up each data type in the XCCDF data model. Note that the properties that comprise a Benchmark or Item are an ordered sequence of property values, and the order in which they appear SHALL determine the order in which they are processed.

Properties with a data type of "text" are string data that MAY include embedded formatting directives and hypertext links. Properties of type "string" SHALL NOT include formatting. Properties of type "identifier" MUST be strings without spaces or formatting, obeying the definition of "NCName" from the XML Schema specification [4].

All lists in this section use the following convention:

Property (Type, Count): Description

Note that a minimum Count value of 0 indicates that the property is OPTIONAL, and a minimum value of 1 or greater indicates that the property is REQUIRED.

5.2.1. Benchmark

The possible properties of a Benchmark are:

- id (identifier, 1): Benchmark identifier
- status (string + date, 1-n): Status of the Benchmark (see below) and date at which it attained that status (at least one status property MUST appear; if several appear, then the one with the latest date SHALL apply)
- title (text, 0-n): Title of the XCCDF Benchmark document
- description (text, 0-n): Text that describes the Benchmark
- version (string + date + URI, 1): Version number of the Benchmark (REQUIRED), with the date and time when the version was completed (REQUIRED) and an update URI [9] (OPTIONAL)
- notice (text, 0-n): Legal notices or copyright statements about this Benchmark; each notice SHALL have a unique identifier and text value

- front-matter (text, 0-n): Text for the front of the Benchmark document
- rear-matter (text, 0-n): Text for the back of the Benchmark document
- reference (special, 0-n): A bibliographic reference for the Benchmark document: metadata or a simple string, plus an OPTIONAL URL
- platform-specification (special, 0-1): A list of complex platform definition, in Common Platform Enumeration (CPE 2.2) language format [5]
- platform (URI, 0-n): Target platforms for this Benchmark, each a URI referring to a platform listed in the community CPE 2.2 dictionary or an identifier defined in the CPE 2.2 Language platform-specification property
- plain-text (string + identifier, 0-n): Reusable text blocks, each with a unique identifier; these MAY be included in other text blocks in the Benchmark
- model (URI + parameters, 0-n): Suggested scoring model or models to be used when computing a compliance score for this Benchmark
- profiles (Profile, 0-n): Profiles that reference and customize sets of Items in the Benchmark
- values (Value, 0-n): Tailoring values that support Rules and descriptions in the Benchmark
- groups (Group, 0-n): Groups that comprise the Benchmark; each group MAY contain additional Values, Groups, and Rules
- rules (Rule, 0-n): Rules that comprise the Benchmark
- test-results (TestResult, 0-n): Benchmark test result records (one per Benchmark run)
- metadata (special, 0-n): Discovery metadata for the Benchmark
- resolved (boolean, 0-1): True if Benchmark has already undergone the resolution process (see Section 5.3)
- style (string, 0-1): Name of a benchmark authoring style or set of conventions to which this Benchmark conforms

- style-href (URI, 0-1): URL of a supplementary stylesheet or schema extension that MAY be used to check conformance to the named style

- signature (special, 0-1): A digital signature asserting authorship and allowing verification of the integrity of the Benchmark

Conceptually, a Benchmark SHOULD contain Group, Rule, and Value objects, and it MAY also contain Profile and TestResult objects. For ease of reading and simplicity of scoping, all Profiles MUST precede all Value objects, which MUST precede all Groups and Rules, which MUST precede all TestResults. These objects SHALL either be directly embedded in the Benchmark or incorporated via W3C standard XML Inclusion [2].

Each status property SHALL consist of a status string and a date. Permissible string values are "accepted", "draft", "interim", "incomplete", and "deprecated". Benchmark authors SHOULD mark their Benchmarks with a status to indicate a level of maturity or consensus. A Benchmark SHOULD contain one or more status properties, each holding a different status value and the data on which the Benchmark reached that status.

Generally, XCCDF items MAY be qualified by platform using Common Platform Enumeration (CPE) Names, as defined in the CPE 2.2 Specification [5]. In CPE, a specific platform is identified by a unique URI. Each Rule, Group, Profile, and the Benchmark itself MAY possess platform properties, each containing a CPE Name URI indicating the hardware or software platform to which the object applies. CPE 2.2 Names can express only unitary or simple platforms (e.g. "cpe:/o:microsoft:windows-nt:xp::pro" for Microsoft Windows XP Professional Edition). Sometimes, XCCDF rules require more complex qualification. The platform-specification property contains a list of one or more complex platform definitions expressed using CPE Language schema. Each definition bears a locally unique identifier. These identifiers MAY be used in platform properties in place of CPE Names.

Note that CPE Names MAY be used in a Benchmark or other objects without defining them explicitly. CPE Names for common IT platforms are generally defined in the community dictionary, and MAY be used directly. Authors MAY use the platform-specification property to define complex platforms and assign them local identifiers for use in the Benchmark.

The Benchmark platform-specification property and platform properties are OPTIONAL. Authors SHOULD use them to identify the systems or products to which their Benchmarks apply.

The plain-text properties allow commonly used text to be defined once and then re-used in multiple text blocks in the Benchmark. Note that each plain-text MUST have a unique id, and that the ids of other Items and plain-text properties MUST NOT collide. This restriction permits easier implementation of document generation and reporting tools.

Benchmark metadata allows authorship, publisher, support, and other information to be embedded in a Benchmark. Metadata SHOULD comply with existing commercial or government metadata specifications, to allow Benchmarks to be discovered and indexed. The XCCDF data model allows multiple metadata properties for a Benchmark; each property SHOULD provide metadata compliant with a different specification. The primary metadata format, which SHOULD appear in all published Benchmarks, is the simple Dublin Core Elements specification, as documented in [12].

The style and style-href properties MAY be used to indicate that a benchmark conforms to a specific set of conventions or constraints. For example, NIST is designing a set of style conventions for XCCDF benchmarks as part of the SCAP initiatives. The style property holds the name of the style (e.g. "SCAP 1.0") and the style-href property holds a reference to a stylesheet or schema that tools can use to test conformance to the style.

Note that a digital signature, if any, SHALL apply only to the Object in which it appears, but after inclusion processing (note: it may be impractical to use inclusion and signatures together). Any digital signature format employed for XCCDF Benchmarks MUST be capable of identifying the signer, storing all information needed to verify the signature (usually, a certificate or certificate chain), and detecting any change to the content of the Benchmark. XCCDF tools that support signatures at all MUST support the W3C XML-Signature standard enveloped signatures [6].

Legal notice text is handled specially, as discussed in Section 5.3.

5.2.2. Item

The possible properties of an Item are:

- id (identifier, 1): Unique object identifier

- title (text, 0-n): Title of the Item (for human readers)
- description (text, 0-n): Text that describes the Item
- warning (text, 0-n): A cautionary note or caveat about the Item
- status (string + date, 0-n): Status of the Item and date at which it attained that status
- version (string + date + URI, 0-1): Version number of the Benchmark (REQUIRED), with the date and time when the version was completed (REQUIRED) and an update URI (OPTIONAL)
- question (string, 0-n): Interrogative text to present to the user during tailoring
- hidden (boolean, 0-1): If this Item should be excluded from any generated documents (default: false)
- prohibitChanges (boolean, 0-1): If tools should prohibit changes to this Item during tailoring (default: false)
- abstract (boolean, 0-1): If true, then this Item is abstract and exists only to be extended (default: false)
- cluster-id (identifier, 0-1): An identifier to be used from a Profile to refer to multiple Groups and Rules
- reference (special, 0-n): A reference to a document or resource where the user can learn more about the subject of this Item: content is Dublin Core metadata or a simple string (REQUIRED), plus a URL (OPTIONAL)
- signature (special, 0-1): Digital signature over this Item

Every Item MAY include one or more status properties. Each status property value represents a status that the Item has reached and the date at which it reached that status. Benchmark authors MAY use status elements to record the maturity or consensus level for Rules, Groups, and Values in the Benchmark. If an Item does not have an explicit status property value given, then its status SHALL be taken to be that of the Benchmark itself. The status property SHALL NOT be inherited.

There are several Item properties that give the Benchmark author control over how Items may be tailored and presented in documents. The ’hidden’ property simply prevents an Item from appearing in

generated documents. For example, an author might set the hidden property on incomplete Items in a draft Benchmark. The 'prohibitChanges' property advises tailoring tools that the Benchmark author does not wish to allow end users to change anything about the Item. Lastly, a value of true for the 'abstract' property denotes an Item intended only for other Items to extend. In most cases, abstract Items SHOULD also be hidden.

The 'cluster-id' property is OPTIONAL, but it MAY be used to provide a means to identify related Value, Group and Rule items throughout the Benchmark. It is OPTIONAL for cluster identifiers to be unique: all the Items with the same cluster identifier belong to the same cluster. A selector in a Profile MAY refer to a cluster, thus making it easier for authors to create and maintain Profiles in a complex Benchmark. The cluster-id property SHALL NOT be inherited.

5.2.2.1. Group::Item

The possible properties of a Group are:

- requires (identifier, 0-n): The id of another Group or Rule in the Benchmark that MUST be selected for this Group to be applied and scored properly

- conflicts (identifier, 0-n): The id of another Group or Rule in the Benchmark that MUST be unselected for this Group to be applied and scored properly

- selected (boolean, 1): If true, this Group SHALL be processed as part of the Benchmark when it is applied to a target system; an unselected Group SHALL NOT be processed, and none of its contents SHALL be processed either (i.e., all descendants of an unselected group SHALL be implicitly unselected). Default is true. MAY be overridden by a Profile.

- rationale (text, 0-n): Descriptive text giving rationale or motivations for abiding by this Group

- platform (URI, 0-n): Platforms to which this Group applies, CPE Names or CPE platform specification identifiers

- cluster-id (identifier, 0-1): An identifier to be used from Benchmark profiles to refer to multiple Groups and Rules

- extends (identifier, 0-1): An id of a Group on which to base this Group

- weight (float, 0-1): The relative scoring weight of this Group, for computing a compliance score; MAY be overridden by a Profile
- values (Value, 0-n): Values that belong to this Group
- groups (Group, 0-n): Sub-groups under this Group
- rules (Rule, 0-n): Rules that belong to this Group

A Group MAY be based on (extend) another Group. The semantics of inheritance work differently for different properties, depending on their allowed count. For Items that belong to a Group, the extending Group SHALL include all the Items of the extended Group, plus any defined inside the extending Group. For any property that is allowed to appear more than once, the extending Group SHALL get the sequence of property values from the extended group, plus any of its own values for that property. For any property that is allowed to appear at most once, the extending Group SHALL get its own value for the property if one appears, otherwise it SHALL get the extended Group's value of that property. Items that belong to an extended group are treated specially: the id property of any Item copied as part of an extended group MUST be replaced with a new, uniquely generated id. A Group for which the abstract property is true exists only to be extended by other Groups; it SHALL NOT appear in a generated document, and none of the Rules defined in it SHOULD be checked in a compliance test. Abstract Group objects SHALL be removed during resolution; for more information, see Section 5.3.

To give the Benchmark author more control over inheritance for extending Groups (and other XCCDF objects), all textual properties that may appear more than once MAY bear an override attribute. For more information about inheritance overrides and extension, see Section 5.3.

The requires and conflicts properties provide a means for Benchmark authors to express dependencies among Rules and Groups. Their exact meaning depends on what sort of processing the Benchmark is undergoing, but in general the following approach SHOULD be applied: if a Rule or Group is about to be processed, and any of the Rules or Groups identified in a requires property have a selected property value of false or any of the Items identified in a conflicts property have a selected property value of true, then processing for the Item SHOULD be skipped and its selected property SHOULD be set to false.

The platform property of a Group indicates that the Group contains platform-specific Items that apply to some set of (usually related)

platforms. First, if a Group does not possess any platform properties, then it SHALL apply to the same set of platforms as its enclosing Group or the Benchmark. Second, for tools that perform compliance checking on a platform, any Group whose set of platform property values do not include the platform on which the compliance check is being performed SHOULD be treated as if their selected property were set to false. Third, the platforms to which a Group apply SHOULD be a subset of the platforms applicable for the enclosing Benchmark. Last, if no platform properties appear anywhere on a Group or its enclosing Group or Benchmark, then the Group nominally SHALL apply to all platforms.

The weight property denotes the importance of a Group relative to its sibling in the same Group or its siblings in the Benchmark (for a Rule that is a child of the Benchmark). Scoring SHALL be computed independently for each collection of sibling Groups and Rules, then normalized as part of the overall scoring process. For more information about scoring, see Section 5.3.

5.2.2.2. Rule::Item

The possible properties of a Rule are:

- selected (boolean, 1): If true, this Rule SHALL be checked as part of the Benchmark when the Benchmark is applied to a target system; an unselected rule SHALL NOT be checked and SHALL NOT contribute to scoring. Default is true. MAY be overridden by a Profile.

- extends (identifier, 0-1): The id of a Rule on which to base this Rule (MUST match the id of another Rule)

- multiple (boolean, 0-1): Whether this rule SHOULD be multiply instantiated. If false, then Benchmark tools SHOULD avoid multiply instantiating this Rule, the default is false

- role (string, 0-1): Rule’s role in scoring and reporting; one of the following: "full", "unscored", "unchecked". Default is "full". MAY be overridden by a Profile.

- severity (string, 0-1): Severity level code, to be used for metrics and tracking. One of the following: "unknown", "info", "low", "medium", "high". Default is "unknown". MAY be overridden by a Profile

- weight (float, 0-1): The relative scoring weight of this Rule, for computing a compliance score. Default is 1.0. MAY be overridden by a Profile

- rationale (text, 0-n): Some descriptive text giving rationale or motivations for complying with this Rule

- platform (URI, 0-n): Platforms to which this Rule applies, CPE Names or CPE platform-specification identifiers

- requires (identifier, 0-n): The id of another Group or Rule in the Benchmark that SHOULD be selected for this Rule to be applied and scored properly

- conflicts (identifier, 0-n): The id of another Group or Rule in the Benchmark that SHOULD be unselected for this Rule to be applied and scored properly

- ident (string + URI, 0-n): A long-term, globally meaningful name for this Rule. MAY be the name or identifier of a security configuration issue or vulnerability that the Rule remediates. Has an associated URI that denotes the organization or naming scheme which assigns the name. (see below)

- impact-metric (string, 0-1): The impact metric for this rule, expressed as a CVSS score. (see below)

- profile-note (text + identifier, 0-n): Descriptive text related to a particular Profile. This property allows a Benchmark author to describe special aspects of the Rule related to one or more Profiles. It has an id that MAY be specified as the 'note-tag' property of a Profile (see the Profile description, below)

- fixtext (special, 0-n): Prose that describes how to fix the problem of non-compliance with this Rule; each fixtext property MAY be associated with one or more fix property values

- fix (special, 0-n): A command string, script, or other system modification statement that, if executed on the target system, can bring it into full, or at least better, compliance with this Rule

- check (special, 0-n): The definition of, or a reference to, the target system check needed to test compliance with this Rule. A check consists of three parts: the checking system specification on which it is based, a list of Value objects to export, and the content of the check itself. If a Rule has several check properties, each MUST employ a different checking system

- complex-check (special, 0-1): A complex check is a boolean expression of other checks. At most one complex-check MAY appear in a Rule (see below)

A Rule MAY be based on (extend) another Rule. This means that the extending Rule SHALL inherit all the properties of the extended or base Rule, some of which it may override with new values. For any property that is allowed to appear more than once, the extending Rule SHALL get the sequence of property values from the extended group, plus any of its own values for that property. For any property that is allowed to appear at most once, the extending Rule SHALL get its own value for the property if one appears, otherwise it SHALL get the extended Rule’s value of that property. A Rule for which the abstract property is true SHOULD NOT be included in any generated document, and MUST NOT be checked in any compliance test. Abstract Rules SHALL be removed during resolution (see Section 5.3).

The ’multiple’ property provides direction about multiple instantiation to a processing tool applying the Rule. By setting ’multiple’ to true, the Rule’s author is directing that separate components of the target to which the Rule can apply SHOULD be tested separately and the results SHOULD be recorded separately. By setting ’multiple’ to false, the author is directing that the test results of such components SHALL be combined. If the processing tool cannot perform multiple instantiation, or if multiple instantiation of the Rule is not applicable for the target system, then processing tools MAY ignore this property.

The role property gives the Benchmark author additional control over Rule processing during application of a Benchmark. The default role ("full") means that the Rule SHALL be checked, SHALL contribute to scoring according to the scoring model, and SHALL appear in any output reports. The "unscored" role means that the Rule SHALL be checked and SHALL appear in any output reports, but SHALL NOT contribute to score computations. The "unchecked" role means that the Rule SHALL NOT be checked, its Rule result status SHALL be set to "notchecked", and it SHALL NOT contribute to scoring, but it MAY appear in output reports. The "unchecked" role is meant primarily for Rules that contain informational text, but for which no automated check is practical.

The weight property denotes the importance of a rule relative to its sibling in the same Group or its siblings in the Benchmark (for a Rule that is a child of the Benchmark). For more information about scoring, see Section 5.3.

The platform properties of a Rule indicate the platforms to which the Rule applies. Each platform property SHALL assert a single CPE Name or a CPE Language identifier. If a Rule does not possess any platform properties, then it SHALL apply to the same set of platforms as its enclosing Group or Benchmark. For tools that perform compliance checking on a platform, if a Rule's set of platform property values does not include the platform on which the compliance check is being performed, the Rule SHOULD be treated as if its selected property were set to false. Any platform property value that appears on a Rule SHOULD be a member of the set of platform property values of the enclosing Benchmark. Finally, if no platform properties appear anywhere on a Rule or its enclosing Group or Benchmark, then the Rule SHALL apply to all platforms.

Each ident property SHALL contain a globally meaningful name in some security domain; the string value of the property is the name, and a Uniform Resource Identifier (URI) [9] designates the scheme or organization that assigned the name. By setting an 'ident' property on a Rule, the Benchmark author effectively declares that the Rule instantiates, implements, or remediates the issue for which the name was assigned. For example, the ident value might be a Common Vulnerabilities and Exposures (CVE) identifier; the Rule would be a check that the target platform was not subject to the vulnerability named by the CVE identifier, and the URI would be that of the CVE Web site.

The impact-metric property contains a multi-part rating of the potential impact of failing to meet this Rule. The string value of the property SHOULD be a base vector expressed according to the Common Vulnerability Scoring System (CVSS) version 2.0 [7].

The check property consists of the following: a selector for use with Profiles, a URI that designates the checking system or engine, a set of export declarations, and the check content. The checking system URI tells a compliance checking tool what processing engine it MUST use to interpret or execute the check. XCCDF also supports conveyance of tailoring values from the XCCDF processing environment down to the checking system, via export declarations. Each export declaration maps an XCCDF Value object id to an external name or id for use by the checking system. The check content is an expression or document in the language of the checking system; it MAY appear

inside the XCCDF document (an enveloped check) or as a reference (a detached check).

In place of a 'check' property, XCCDF allows a 'complex-check' property. A complex check is a boolean expression whose individual terms are checks or complex-checks. This allows Benchmark authors to re-use checks in more flexible ways, and to mix checks written with different checking systems. A Rule MAY have at most one 'complex-check' property; on inheritance, the extending Rule's complex-check SHALL replace the extended Rule's complex-check. If both check properties and a complex-check property appear in a Rule, then the check properties MUST be ignored. The following operators are allowed for combining the constituents of a complex-check:

- AND - if and only if all terms evaluate to Pass (true), then the complex-check SHALL evaluate to Pass.

- OR - if any term evaluates to Pass, then the complex-check SHALL evaluate to Pass.

Truth-tables for the operators appear under their detailed descriptions in the next section. Note that each complex-check MAY also specify that the expression should be negated (boolean not).

The properties fixtext and fix exist to allow a Benchmark author to specify a way to remediate non-compliance with a Rule. The 'fixtext' property provides a prose description of the fix that needs to be made; in some cases this may be all that is possible to do in the Benchmark (e.g., if the fix requires manipulation of a GUI or installation of additional software). The 'fix' property provides a direct means of changing the system configuration to accomplish the necessary change (e.g., a sequence of command-line commands; a set of lines in a system scripting language like Bourne shell or in a system configuration language like Windows INF format; a list of update or patch ID numbers).

The fix and fixtext properties may be used by tools to support more sophisticated facilities for automated and interactive remediation of Benchmark findings. The following attributes MAY be associated with a fix or fixtext property value:

- strategy - a keyword that denotes the method or approach for fixing the problem. This applies to both fix and fixtext. Permitted values: unknown (default), configure, combination, disable, enable, patch, policy, restrict, update.

- disruption - an estimate for how much disruption the application of this fix will impose on the target. This applies to fix and fixtext. Permitted values: unknown, low, medium, high.

- reboot - whether or not remediation will require a reboot or hard reset of the target. This applies to fix and fixtext. Permitted values: true (1) and false (0).

- system - a URI representing the scheme, language, engine, or process for which the fix contents are written. See Appendix A for several general-purpose URNs for this, but it is expected that tool vendors and system providers may need to define target-specific ones. This applies to fix only.

- id/fixref - these attributes will allow fixtext properties to be associated with specific fix properties (pair up explanatory text with specific fix procedures).

- platform - in case different fix scripts or procedures are required for different target platform types (e.g., different patches for Windows 2000 and Windows XP), this attribute allows a CPE Name or CPE Language definition to be associated with a fix property.

For more information, consult the definitions of the fix and fixtext elements in Section 6.2.

5.2.2.3. Value::Item

The possible properties of a Value are:

- value (string + id, 1-n): The current value of this Value

- default (string + id, 0-n): Default value of this Value object

- type (string, 0-1): The data type of the Value: "string", "number", or "boolean" (default: "string")

- extends (identifier, 0-1): The id of a Value on which to base this Value

- operator (string, 0-1): The operator to be used for comparing this Value to some part of the test system’s configuration (see list below)

- lower-bound (number + identifier, 0-n): Minimum legal value for this Value (applies only if type is ’number’)

- upper-bound (number + identifier, 0-n): Maximum legal value for this Value (applies only if type is 'number')
- choices (list + id, 0-n): A list of legal or suggested values for this Value object, to be used during tailoring and document generation
- match (string [regular expr.], 0-n): A regular expression which the Value MUST match to be legal (for more information, see [8])
- interactive (boolean, 0-1): Tailoring for this Value should also be performed during Benchmark application (default is false)
- interfaceHint (string, 0-1): User interface recommendation for tailoring
- source (URI, 0-n): URI indicating where the Benchmark tool may acquire a value for this Value object

A Value is content that can be substituted into properties of other Items, including the interior of structured check specifications and fix scripts. A tool MAY choose any convenient form to store a Value's value property, but the data type conveys how the value should be treated during Benchmark compliance testing. The data type property MAY also be used to give additional guidance to the user or to validate the user's input. For example, if a Value object's type property was "number", then a tool might choose to reject user tailoring input that was not composed of digits. The default property holds a default value for the value property; tailoring tools MAY present the default value to users as a suggestion.

A Value object MAY extend another Value object. In such cases, the extending object SHALL receive all the properties of the extended object, and MAY override them where needed. A Value object with the abstract property true SHALL NOT be included in any generated document, and MAY NOT be exported to any compliance checking engine.

When defining a Value object, the Benchmark author MAY specify the operator to be used for checking compliance with the value. For example, one part of an OS Benchmark might be checking that the configuration included a minimum password length; the Value object that holds the tailorable minimum could have type "number" and operator "greater than". Exactly how Values are used in rules may depend on the capabilities of the checking system. Tailoring tools and document generation tools MAY ignore the 'operator' property; therefore, Benchmark authors SHOULD included sufficient information in the description and question properties to make the role of the

Value clear. The table below describes the operators permitted for each Value type.

Value Type	Available Operators	Remarks
number	equals, not equal, less than, greater than, less than or equal, greater than or equal	Default operator: equals
boolean	equals, not equal	Default operator: equals
string	equals, not equal, pattern match (pattern match means regular expression match; should comply with [8])	Default operator: equals

A Value object includes several properties that constrain or limit the values that the Value may be given: value, default, match, choices, upper-bound, and lower-bound. Benchmark authors MAY use these Value properties to assist users in tailoring the Benchmark. These properties MAY appear more than once in a Value, and MAY be marked with a selector tag id. At most one instance of each MAY omit its selector tag. For more information about selector tags, see the description of the Profile object below.

The upper-bound and lower-bound properties constrain the choices for Value items with a type property of ’number’. For any other type, they are meaningless. The bounds they indicate are always inclusive. For example, if the lower-bound property for a Value is given as "3", then 3 is a legal value.

The ’choices’ property holds a list of one or more particular values for the Value object; the ’choices’ property also bears a boolean flag, ’mustMatch’, which indicates that the enumerated choices are the only legal ones (mustMatch="1") or that they are merely suggestions (mustMatch="0"). The choices property SHOULD be used when there are a moderate number of known values that are most appropriate. For example, if the Value were the authentication mode for a server, the choices might be "password" and "pki".

The match property provides a regular expression pattern that a tool MAY apply, during tailoring, to validate user input. The ’match’ property SHALL apply only when the Value type is ’string’ or ’number’. For example, if the Value type was ’string’, but the value

was meant to be a Cisco IOS router interface name, then the Value match property might be set to "[A-Za-z]+ *[0-9]+(/[0-9.]+)*". This would allow a tailoring tool to reject an invalid user input like "f8xq+" but accept a legal one like "Ethernet1/3".

If a Value's prohibitChanges property is set to true, then it means that the Value's value SHALL NOT be changed by the user. This might be used by Benchmark authors in defining values that are integral to compliance, such as a timeout value, or it might be used by enterprise security officers in constraining a Benchmark to more tightly reflect organizational or site security policies. (In the latter case, a security officer could use the extension facility to make an untailorable version of a Value object, without rewriting it.) A Value object MAY have a 'hidden' property; if the hidden property is true, then the Value SHOULD NOT appear in a generated document, but its value MAY still be used.

If the interactive property is set, it is a hint to the Benchmark checking tool to ask the user for a new value for the Value at the beginning of each application of the Benchmark. The checking tool MAY ignore the property if asking the user is not feasible or not supported. Similarly, the 'interfaceHint' property allows the Benchmark author to supply a hint to a benchmarking or tailoring tool about how the user might select or adjust the Value. The following strings are valid for the 'interfaceHint' property: "choice", "textline", "text", "date", and "datetime".

The source property allows a Benchmark author to supply a URI, possibly tool-specific, that indicates where a benchmarking or tailoring tool may acquire values, value bounds, or value choices.

5.2.3. Profile

The possible properties for a Profile are:

- id (identifier, 1): Unique identifier for this Profile
- title (string, 1-n): Title of the Profile, for human readers
- description (text, 0-n): Text that describes the Profile
- extends (identifier, 0-1): The id of a Profile on which to base this Profile
- abstract (boolean, 0-1): If true, then this Profile exists solely to be extended by other Profiles, and SHALL NOT be applied to a Benchmark directly (default: false)

- note-tag (identifier, 0-1): Tag identifier to match profile-note properties in Rules
- status (string + date, 0-n): Status of the Profile and date at which it attained that status
- version (string + date, 0-1): Version of the Profile, with timestamp and update URI
- prohibitChanges (boolean, 0-1): Whether or not tools SHOULD prohibit changes to this Profile (default: false)
- platform (URI, 0-n): A target platform for this Profile, a CPE Name or platform-specification identifier. Multiple platform URIs MAY be listed if the Profile applies to several platforms
- reference (string + URL, 0-n): A reference to a document or resource where the user can learn more about the subject of this Profile: a REQUIRED string and OPTIONAL URL
- selectors (special, 0-n): References to Groups, Rules, and Values, see below (references MAY be the unique id of an Item, or a cluster id)
- signature (special, 0-1): Digital signature over this Profile

A Profile object is a named tailoring of a Benchmark. While a Benchmark can be tailored in place, by setting properties of various objects, only Profiles allow one Benchmark document to hold several independent tailorings.

A Profile MAY extend another Profile in the same Benchmark. The set of platform, reference, and selector properties of the extended Profile are prepended to the list of properties of the extending Profile. Inheritance of title, description, and reference properties are handled in the same way as for Rule objects.

The note-tag property is a simple identifier. It specifies which profile-note properties on Rules are to be associated with this Profile.

Benchmark authors MAY use the Profile’s ’status’ property to record the maturity or consensus level of a Profile. If the status is not given explicitly in a Profile definition, then the Profile SHALL have the same status as its parent Benchmark. Note that status properties SHALL NOT be inherited.

Each Profile contains a list of selectors which express a particular customization or tailoring of the Benchmark. There are four kinds of selectors:

- select - a Rule/Group selector. This selector designates a Rule, Group, or cluster of Rules and Groups. It overrides the selected property on the designated Items. It provides a means for including or excluding rules from the Profile.

- set-value - a Value selector. This selector overrides the value property of a Value object, without changing any of its other properties. It provides a means for directly specifying the value of a variable to be used in compliance checking or other Benchmark processing. This selector MAY also be applied to the Value items in a cluster, in which case it overrides the value properties of all of them.

- refine-rule - a Rule/Group selector. This selector allows the Profile author to override the scoring weight, severity, and role of a Rule, Group, or cluster of Rules and Groups. Despite the name, this selector does apply for Groups, but only to their weight property.

- refine-value - a Value selector. This selector designates the Value constraints to be applied during tailoring, for a Value object or the Value members of a cluster. It provides a means for authors to impose different constraints on tailoring for different profiles. (Constraints MUST be designated with a selector id. For example, a particular numeric Value might have several different sets of 'value', 'upper-bound', and 'lower-bound' properties, designated with different selector ids. The refine-value selector tells benchmarking tools which value to employ and bounds to enforce when that particular profile is in effect.)

All of the selectors except set-value MAY include remark elements, to allow the benchmark author to add explanatory material to individual elements of the Profile.

Selectors SHALL be applied in the order they appear within the Profile. For selectors that refer to the same Item or cluster, this means that later selectors MAY override or change the actions of earlier ones.

5.2.4. TestResult

The possible properties of TestResult are:

- id (identifier, 1): Identifier for this TestResult object
- benchmark (URI, 0-1): Reference to Benchmark; REQUIRED if this TestResult object is in a file by itself, OPTIONAL otherwise
- version (string, 0-1): The version number string copied from the Benchmark
- title (string, 0-n): Title of the test, for human readers
- remark (string, 0-n): A remark about the test, possibly supplied by the person administering the Benchmark checking run
- organization (string, 0-n): The name of the organization or enterprise responsible for applying this Benchmark and generating this result
- identity (string + boolean, 0-1): Information about the system identity employed during application of the Benchmark
- start-time (timestamp, 0-1): Time when test began
- end-time (timestamp, 1): Time when test was completed and the results recorded
- test-system (string, 0-1): Name of the test tool or program that generated this TestResult object; SHOULD be a CPE 2.2 Name [5]
- target (string, 1-n): Name of the target system whose test results are recorded in this object
- target-address (string, 0-n): Network address of the target
- target-facts (special, 0-1): A sequence of named facts about the target system or platform, including a type qualifier
- platform (URI, 0-n): The CPE platform URI indicating the platforms which the target system was found to meet. Tools MAY insert multiple platform URIs if the target system met multiple relevant platform definitions
- profile (identifier, 0-1): The identifier of the Benchmark profile used for the test, if any
- set-value (string + id, 0-n): Specific settings for Value objects used during the test, one for each Value

- rule-results (special, 1-n): Outcomes of individual Rule tests, one per Rule instance
- score (float + URI, 1-n): An overall score for this Benchmark test; at least one MUST appear
- signature (special, 0-1): Digital signature over this TestResult object

A TestResult object represents the results of a single application of the Benchmark to a single target platform. The properties of a TestResult object include test time, the identity and other facts about the system undergoing the test, and Benchmark information. If the test was conducted using a specific Profile of the Benchmark, then a reference to the Profile MAY be included. Also, multiple set-value properties MAY be included, giving the identifier and value for the Values that were used in the test. The 'test-system' property gives the CPE 2.2 Name for the testing tool or application responsible for generating this TestResult object.

At least one target property MUST appear in the TestResult object. Each appearance of the property supplies a name by which the target host or device was identified at the time the test was run. The name MAY be any string, but applications SHOULD include the fully qualified Domain Name System (DNS) name whenever possible. The 'target-address' property is OPTIONAL; each appearance of the property supplies an address which was bound by the target at the time the test was run. Typical forms for the address include: Internet Protocol version 4 (IPv4) address, Internet Protocol version 6 (IPv6) address, and Ethernet media access control (MAC) address.

The 'organization' property documents the organization, enterprise, or group responsible for the benchmark. The property MAY appear multiple times, to indicate multiple levels of an organizational hierarchy, in which case the highest-level organization SHOULD appear first, followed by subordinate organizations.

The identity property provides up to three pieces of information about the system identity used to apply the benchmark and generate the findings encapsulated by this TestResult object. The three pieces of information are:

- authenticated - whether the identity was authenticated with the target system during the application of the benchmark [boolean].

- privileged - whether the identity was granted privileges beyond those of a normal system user, such as superuser on Unix or LocalSystem rights on Windows [boolean].

- name - the name of the authenticated identity [string]. (The names of privileged identities are considered sensitive for most systems. Therefore, this part of the identity property MAY be omitted.)

The target-facts list is an OPTIONAL part of the TestResult object. It contains a list of one or more individual facts about the target system or platform. Each fact consists of the following: a name (URI), a type ("string", "number", or "boolean"), and the value of the fact itself.

The main content of a TestResult object is a collection of rule-result records, each giving the result of a single instance of a rule application against the target. The TestResult MUST include one rule-result record for each Rule that was selected in the resolved Benchmark; it MAY also include rule-result records for Rules that were unselected in the Benchmark. A rule-result record contains the properties listed below. For more information about applying and scoring Benchmarks, see Section 5.3.5.

5.2.4.1. TestResult/rule-result

The possible properties of a rule-result are:

- rule-idref (identifier, 1): Identifier of a Benchmark Rule (from the Benchmark designated in the TestResult)

- time (timestamp, 0-1): Time when application of this instance of this Rule was completed

- version (string, 0-1): The version number string copied from the version property of the Rule

- severity (string, 0-1): The severity string code copied from the Rule; defaults to "unknown"

- ident (string + URI, 0-n): A globally meaningful name and URI for the issue or vulnerability, copied from the Rule

- result (string, 1): Result of this test: one of status values listed below

- override (special, 0-n): An XML block explaining how and why an auditor chose to override the Rule's result status

- instance (string, 0-n): Name of the target system component to which this result applies, for multiply instantiated Rules. MAY also include context and hierarchy information for nested contexts (see below for details)

- message (string + code, 0-n): Diagnostic messages from the checking engine (REQUIRED), with OPTIONAL severity (this would normally appear only for result values of "fail" or "error")

- fix (string, 0-1): Fix script for this target platform, if available (would normally appear only for result values of "fail")

- check (special, 0-n): Encapsulated or referenced results to detailed testing output from the checking engine (if any); if multiple checks were executed as part of a complex-check, then data for each MAY appear here

The result of a single test SHALL be one of the following:

- pass - the target system or system component satisfied all the conditions of the Rule; a pass result contributes to the weighted score and maximum possible score. [Abbreviation: P]

- fail - the target system or system component did not satisfy all the conditions of the Rule; a fail result contributes to the maximum possible score. [Abbreviation: F]

- error - the checking engine encountered a system error and could not complete the test, therefore the status of the target's compliance with the Rule is not certain. This could happen, for example, if a Benchmark testing tool were run with insufficient privileges. [Abbreviation: E]

- unknown - the testing tool encountered some problem and the result is unknown. For example, a result of 'unknown' might be given if the Benchmark testing tool were unable to interpret the output of the checking engine. [Abbreviation: U]

- notapplicable - the Rule was not applicable to the target of the test. For example, the Rule might have been specific to a different version of the target OS, or it might have been a test against a platform feature that was not installed. Results with this status SHALL NOT contribute to the Benchmark score. [Abbreviation: N]

- notchecked - the Rule was not evaluated by the checking engine. This status is designed for Rules with a role of "unchecked", and for Rules that have no check properties. It may also correspond to a status returned by a checking engine. Results with this status SHALL NOT contribute to the Benchmark score. [Abbreviation: K]

- notselected - the Rule was not selected in the Benchmark. Results with this status SHALL NOT contribute to the Benchmark score. [Abbreviation: S]

- informational - the Rule was checked, but the output from the checking engine is simply information for auditor or administrator; it is not a compliance category. This status is the default for Rules with a role of "unscored". This status value is designed for Rules whose main purpose is to extract information from the target rather than test compliance. Results with this status SHALL NOT contribute to the Benchmark score. [Abbreviation: I]

- fixed - the Rule had failed, but was then fixed (possibly by a tool that can automatically apply remediation, or possibly by the human auditor). Results with this status SHOULD be scored the same as pass. [Abbreviation: X]

The instance property specifies the name of a target subsystem or component that passed or failed a Rule. This is important for Rules that apply to components of the target system, especially when a target might have several such components. For example, a Rule might specify a particular setting that needs to be applied on every interface of a firewall; for Benchmark compliance results, a firewall target with three interfaces would have three rule-result elements with the same rule id, each with an independent value for the ’result’ property. For more discussion of multiply instantiated Rules, see Section 5.3.7.

The ’check’ property consists of the URI that designates the checking system, and detailed output data from the checking engine. The detailed output data MAY take the form of encapsulated XML or text data, or it MAY be a reference to an external URI. (Note: this

is analogous to the form of the Rule object's check property, used for referring to checking engine input.)

The override property provides a mechanism for an auditor to change the Rule result assigned by the Benchmark checking tool. This is necessary (a) when checking a rule requires reviewing manual procedures or other non-IT conditions, and (b) when a Benchmark check gives an inaccurate result on a particular target system. The override element contains the following properties:

- time (timestamp, 1): When the override was applied
- authority (string, 1): Name or other identification for the human principal authorizing the override
- old-result (string, 1): The rule result status before this override
- new-result (string, 1): The new, override rule result status
- remark (string, 1): Rationale or explanation text for why or how the override was applied

XCCDF is not intended to be a database format for detailed results; the TestResult object offers a way to store the results of individual tests in modest detail, with the ability to reference lower-level testing data.

5.3. Processing Models

The XCCDF specification is designed to support automated XCCDF document processing by a variety of tools. There are five basic types of processing that a tool might apply to an XCCDF document:

1. Tailoring. This type of processing involves loading an XCCDF document, allowing a user to set the value property of Value items and the selected property of all Items, and then generating a tailored XCCDF output document.
2. Document Generation. This type of processing involves loading an XCCDF document and generating textual or formatted output, usually in a form suitable for printing or human perusal.

3. Transformation. This is the most open-ended of the processing types: it involves transforming an XCCDF document into a document in some other representation. Typically, a transformation process will involve some kind of stylesheet or specification that directs the transformation (e.g., an Extensible Stylesheet Language Transformation (XSLT) stylesheet). This kind of processing can be used in a variety of contexts, including document generation.

4. Compliance Checking. This is the primary form of processing for XCCDF documents. It involves loading an XCCDF document, checking target systems or data sets that represent the target systems, computing one or more scores, and generating one or more XCCDF TestResult objects. Some tools might also generate other outputs or store compliance information in some kind of database.

5. Test Report Generation. This form of processing can be performed only on an XCCDF document that includes one or more TestResult objects. It involves loading the document, traversing the list of TestResult objects, and generating non-XCCDF output and/or human-readable reports about selected ones.

Tailoring, document generation, and compliance checking all share a similar processing model consisting of two steps: loading and traversal. The processing sequence required for loading is described in the subsection below. Note that loading MUST be complete before traversal begins. When loading is complete, a Benchmark is said to be resolved.

5.3.1. Loading Processing Sequence

Before any loading begins, a tool SHOULD initialize an empty set of legal notices and an empty dictionary of object ids. The following is a list of the sub-steps that SHALL be followed for loading processing:

1. Loading.Import - Import the XCCDF document into the program and build an initial internal representation of the Benchmark object, Groups, Rules, and other objects. If the file cannot be read or parsed, then Loading fails. (At the beginning of this step, any inclusion processing specified with XInclude elements SHOULD be performed. The resulting XML information set SHOULD be validated against the XCCDF schema.) Go to the next step: Loading.Noticing.

2. Loading.Noticing - For each notice property of the Benchmark object, add the notice to the tool's set of legal notices. If a notice with an identical id value is already a member of the set, then replace it. If the Benchmark's resolved property is set, then Loading succeeds, otherwise go to the next step: Loading.Resolve.Items.

3. Loading.Resolve.Items - For each Item in the Benchmark that has an extends property, resolve it by using the following steps: (1) if the Item is Group, resolve all the enclosed Items, (2) resolve the extended Item, (3) prepend the property sequence from the extended Item to the extending Item, (4) if the Item is a Group, assign values for the id properties of Items copied from the extended Group, (5) remove duplicate properties and apply property overrides, and (6) remove the extends property. If any Item's extends property identifier does not match the identifier of a visible Item of the same type, then Loading fails. If the directed graph formed by the extends properties includes a loop, then Loading fails. Otherwise, go to the next step: Loading.Resolve.Profiles.

4. Loading.Resolve.Profiles - For each Profile in the Benchmark that has an extends property, resolve the set of properties in the extending Profile by applying the following steps: (1) resolve the extended Profile, (2) prepend the property sequence from the extended Profile to that of the extending Profile, (3) remove all but the last instance of duplicate properties. If any Profile's extends property identifier does not match the identifier of another Profile in the Benchmark, then Loading fails. If the directed graph formed by the extends properties of Profiles includes a loop, then Loading fails. Otherwise, go to Loading.Resolve.Abstract.

5. Loading.Resolve.Abstract - For each Item in the Benchmark for which the abstract property is true, remove the Item. For each Profile in the Benchmark for which the abstract property is true, remove the Profile. Go to the next step: Loading.Resolve.Finalize.

6. Loading.Resolve.Finalize - Set the Benchmark resolved property to true; Loading succeeds.

If the Loading step succeeds for an XCCDF document, then the internal data model should be complete, and every Item should contain all of its own content. An XCCDF file that has no extends properties is called a resolved document. Only resolved XCCDF documents SHOULD be subjected to Transformation processing.

XML Inclusion processing MUST happen before any validation or processing. Typically, it SHOULD be performed by the XML parser as the XML file is processed at the beginning of Loading.Import. XML Inclusion processing is independent of all XCCDF processing.

During the Loading.Resolve.Items and Loading.Resolve.Profiles steps, the processor MUST flatten inheritance relationships. The conceptual model for XCCDF object properties is a list of name-value pairs; property values defined in an extending object are appended to the list inherited from the extending object.

There are five different inheritance processing models for Item and Profile properties:

- None - the property value or values are not inherited.
- Prepend - the property values are inherited from the extended object, but values on the extending object come first, and inherited values follow.
- Append - the property values are inherited from the extended object; additional values may be defined on the extending object.
- Replace - the property value is inherited; a property value explicitly defined on the extending object replaces an inherited value.
- Override - the property values are inherited from the extended object; additional values may be defined on the extending object. An additional value can override (replace) an inherited value, if explicitly tagged as 'override'.

The table below shows the inheritance processing model for each of the properties supported on Group, Rule, Value, and Profile objects.

Processing Model	Properties	Remarks
None	abstract, cluster-id, extends, id, signature, status	These properties cannot be inherited at all; they MUST be given explicitly
Prepend	source, choices	
Append	requires, conflicts, ident, fix, value, default,	Additional rules MAY apply during Benchmark

	operator, lower-bound, upper-bound, match, select, note-tag, refine-value, refine-rule, set-value	processing, tailoring, or report generation
Replace	hidden, prohibitChanges, selected, version, weight, operator, interfaceHint, check, complex-check, role, severity, type, interactive, multiple	For the check property, checks from different systems are considered different properties
Override	title, description, platform, question, rationale, warning, reference, fixtext, profileNote	For properties that have a locale (xml:lang specified), values with different locales are considered to be different properties

Every resolved document MUST satisfy the condition that every id attribute is unique. Therefore, it is very important that the Loading.Resolution step MUST generate a fresh unique id for any Group, Rule, or Value object that gets created through extension of its enclosing Group. One way to do this would be to generate and assign a random unique id during sub-step (4) of Loading.Resolve.Items. Also note that an extends property SHALL be assigned to the newly created Items, based on the id or extends property of the Item that was copied (if the Item being copied has an extends property, then the new Item SHALL get the same value for the extends property, otherwise the new Item SHALL get the id value of the Item being copied as its extends property).

5.3.2. Traversal Processing Sequence

The second step of processing is Traversal. The concept behind Traversal is basically a pre-order, depth-first walk through all the Items that make up a Benchmark. However, Traversal works slightly differently for each of the three kinds of processing, as described further below.

5.3.2.1. Benchmark Processing Algorithm

The id of a Profile MAY be specified as input for Benchmark processing. The following is a list of the sub-steps that SHALL be followed in Benchmark processing:

1. Benchmark.Front - Process the properties of the Benchmark object

2. Benchmark.Profile - If a Profile id was specified, then apply the settings in the Profile to the Items of the Benchmark

3. Benchmark.Content - For each Item in the Benchmark object's items property, initiate Item.Process

4. Benchmark.Back - Perform any additional processing of the Benchmark object properties

The sub-steps Front and Back will be different for each kind of processing, and each tool MAY perform specialized handling of Benchmark properties. For document generation, Profiles MAY be processed separately as part of Benchmark.Back, to generate part of the output document.

5.3.2.2. Item Processing Algorithm

The following is a list of the sub-steps that SHALL be followed for Item processing:

1. Item.Process - Check the contents of the requires and conflicts properties, and if any required Items are unselected or any conflicting Items are selected, then set the selected and allowChanges properties to false.

2. Item.Select - If any of the following conditions holds, cease processing of this Item.

- The processing type is Tailoring, and the optional property and selected property are both false.

- The processing type is Document Generation, and the hidden property is true.

- The processing type is Compliance Checking, and the selected property is false.

- The processing type is Compliance Checking, and the current platform (if known by the tool) is not a member of the set of platforms for this Item.

3. Group.Front - If the Item is a Group, then process the properties of the Group.

4. Group.Content - If the Item is a Group, then for each Item in the Group’s items property, initiate Item.Process.

5. Rule.Content - If the Item is a Rule, then process the properties of the Rule.

6. Value.Content - If the Item is a Value, then process the properties of the Value.

Processing the properties of an Item is the core of Benchmark processing. The list below describes some of the processing in more detail.

- For Tailoring, the key to processing is to query the user and incorporate the user’s response into the data. For a Group or Rule, the user SHOULD be given a yes/no choice if the optional property is true. For a Value item, the user SHOULD be given a chance to supply a string value, possibly validated using the type property. The output of a tailoring tool will usually be another XCCDF file.

- For Document Generation, the key to processing is to generate an output stream that can be formatted as a readable or printable document. The exact formatting discipline will depend on the tool and the target output format. In general, the selected and optional properties are not germane to Document Generation. The platform properties MAY be used during Document Generation for generation of platform-specific versions of a document.

- For Compliance Checking, the key to processing is applying the Rule checks to the target system or collecting data about the target system. Tools will vary in how they do this and in how they generate output reports. It is also possible that some Rule checks MAY need to be applied to multiple contexts or features of the target system, generating multiple pass or fail results for a single Rule object.

Note that it is possible (but inadvisable) for a Benchmark author to set up circular dependencies or conflicts using the requires and conflicts properties. Authors SHOULD NOT do this. To prevent ambiguity, tools MUST process the Items of the Benchmark in order, and MUST NOT change the selected property of any Rule or Group more than once during a processing session.

5.3.3. Substitution Processing

XCCDF supports the notion of named parameters, Value objects, which can be set by a user during the tailoring process, and then substituted into content specified elsewhere in the Benchmark. XCCDF also supports the notion of plain-text definitions in a Benchmark; these are re-usable chunks of text that MAY be substituted into other texts using the substitution facilities described here.

As described in the next section, a substitution SHALL be indicated by a reference to the id of a particular Value object, plain-text definition, or other Item in the Benchmark.

During Tailoring and Document Generation, a tool SHOULD substitute the title property of the Value object for the reference in any text shown to the user or included in the document. At the tool author's discretion, the title MAY be followed by the Value object's value property, suitably demarcated. For plain-text definitions, any reference to the definition SHOULD be replaced by the string content of the definition.

Any appearance of the instance element in the content of a fix element SHOULD be replaced by a locale-appropriate string to represent a target system instance name.

During Compliance Checking, Value objects designated for export to the checking system are passed to it. In general, the interface between the XCCDF checking tool and the underlying checking system or engine MUST support passing the following properties of the Value: value, type, and operator.

During creation of TestResult objects on conclusion of Compliance Checking, any fix elements present in applied Rules, and matching the platform to which the compliance test was applied, SHOULD be subjected to substitution and the resulting string SHOULD be used as the value of the fix element for the rule-result element. Each sub element SHOULD be replaced by the value of the referenced Value object or plain-text definition actually used during the test. Each instance element SHOULD be replaced by the value of the rule-result instance element.

5.3.4. Rule Application and Compliance Scoring

When a Benchmark compliance checking tool performs a compliance run against a system, it accepts as inputs the state of the system and a Benchmark, and produces some outputs, as shown below.

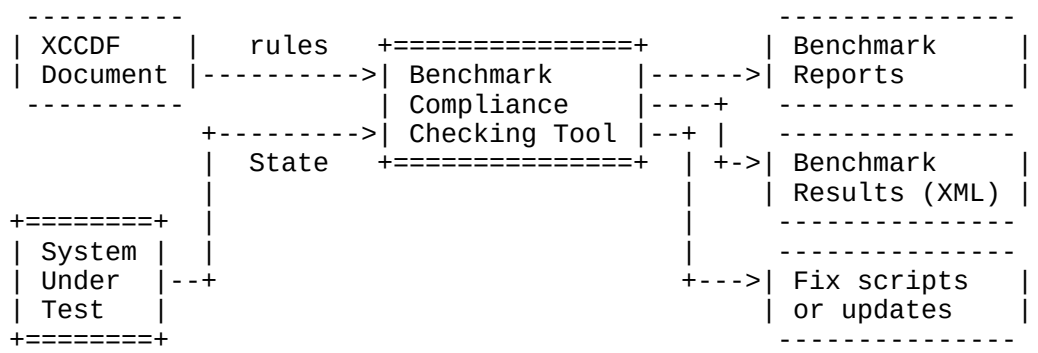


Figure 3 - Workflow for Checking Benchmark Compliance

- Benchmark Report - A human-readable report about compliance, including the compliance score, and a listing of which rules passed and which failed on the system. If a given rule applies to multiple parts or components of the system, then multiple pass/fail entries MAY appear on this list; multiply-instantiated rules are discussed in more detail below. The report MAY also include recommended steps for improving compliance. The format of the benchmark report is not specified here, but MAY be some form of formatted or rich text (e.g., HTML).

- Benchmark results - A machine-readable file about compliance, meant for storage, long-term tracking, or incorporation into other reports (e.g., a site-wide compliance report). This file MAY be in XCCDF, using the TestResult object, or it MAY be in some tool-specific data format.

- Fix scripts - Machine-readable files, usually text, the application of which will remediate some or all of the non-compliance issues found by the tool. These scripts MAY be included in XCCDF TestResult objects.

5.3.5. Scoring and Results Model

Semantically, the output or result of a single Benchmark compliance test consists of four parts:

1. Rule result list - a vector V of result elements e, with each element a 6-tuple e={r, p, I, t, F,O} where:

- r is the Rule id

- p is the test result, one of {pass, fail, error, unknown, notapplicable, notchecked, notselected, informational, fixed}. A test whose result p is 'error' or 'unknown' is treated as 'fail' for the purposes of scoring; tool developers MAY alert the user to erroneous and unknown test results. A test whose result p is one of {notapplicable, notchecked, informational, notselected} SHALL NOT contribute to scoring in any way. A test whose result p is 'fixed' SHALL be treated as a pass for score computation.

- I is the instance set, identifying the system components, files, interfaces, or subsystems to which the Rule was applied. Each element of I is a triple {n,c,p}, where n is the instance name, c is the optional instance context, and p is the optional parent context. The context c, when present, describes the scope or significance of the name n. The parent context p allows the members of I to express nested structure. I MUST be an empty set for tests that are not the result of multiply instantiated Rules (see below).

- t is the time at which the result of the Rule application was decided.

- F is the set of fixes, from the Rule's fix properties, that should bring the target system into compliance (or at least closer to compliance) with the rule. F MAY be null if the Rule did not possess any applicable fix properties, and MUST be null when p is equal to pass. Each fix f in F consists of all the properties defined in the description of the Rule fix property: content, strategy, disruption, reboot, system, id, and platform.

- O is the set of overrides, each o in O consisting of the five properties listed for the rule-result override property: time, authority, old-result, new-result, and remark. Overrides SHALL NOT affect score computation.

2. Scores - a vector S, consisting of one or more score values s, with each s a pair consisting of a real number and a scoring model identifier.

3. Identification - a vector of strings which identify the Benchmark, Profile (if any), and target system to which the Benchmark was applied.

4. Timestamps - two timestamps recording the beginning and the end of the interval when the Benchmark was applied and the results compiled.

Each element of the pass/fail list V conveys the compliance of the system under test, or one component of it, with one Rule of the Benchmark. Each Rule MAY have a weight, title, and other attributes as described above. Each element of V MAY include an instance name, which gives the name of a system component to which the pass or fail designation applies.

XCCDF 1.1.4 defines a default scoring model and three optional scoring models, and also permits Benchmark checking tools to support additional proprietary or community models. A Benchmark MAY specify the scoring model to be used. In the absence of an explicit scoring model specified in the Benchmark, compliance checking tools MUST compute a score based on the default XCCDF model, and MAY compute additional scoring values based on other models. The default model computes a score based on relative weights of sibling rules, as described in the next sub-section.

The fix scripts are collected from the fix properties of the rules in elements of V where p is False. A compliance checking or remediation tool MAY choose to concatenate, consolidate, and/or deconflict the fix scripts; mechanisms for doing so are outside the scope of this specification. In the simplest cases, tools MUST perform Value substitution on each rule’s fix property before making it part of the output results.

5.3.6. Score Computation Algorithms

This sub-section describes the XCCDF default scoring model, which compliance checking tools MUST support, and two additional models that tools MAY support. Each scoring model is identified by a URI. When a Benchmark compliance test is performed, the tool performing the Benchmark MAY use any score computation model designated by the user. The Benchmark author MAY suggest or recommend scoring models by indicating them in the Benchmark object using the "model" property. The default model is indicated implicitly for all Benchmarks.

5.3.6.1. The Default Model

This model is identified by the URI "urn:xccdf:scoring:default". It was the only model supported in XCCDF 1.0, and remains the default for compatibility.

In the default model, computation of the XCCDF score proceeds independently for each collection of siblings in each Group, and then for the siblings within the Benchmark. This relative-to-siblings weighted scoring model is designed for flexibility and to

foster independent authorship of collections of Rules. Benchmark authors should keep the model in mind when assigning weights to Groups and Rules. For a very simple Benchmark consisting only of Rules and no Groups, weights MAY be omitted.

The objects of an XCCDF Benchmark form the nodes of a tree. The default model score computation algorithm simply computes a normalized weighted sum at each tree node, omitting Rules and Groups that are not selected, and Groups that have no selected Rules under them. The algorithm that SHALL be followed at each selected node is:

1. Score.Rule - If the node is a Rule, then assign a count of 1, and if the test result is 'pass', assign the node a score of 100, otherwise assign a score of 0.

2. Score.Group.Init - If the node is a Group or the Benchmark, assign a count of 0, a score s of 0.0, and an accumulator a of 0.0.

3. Score.Group.Recurse - For each selected child of this Group or Benchmark, do the following: (1) compute the count and weighted score for the child using this algorithm, (2) if the child's count value is not 0, then add the child's weighted score to this node's score s, add 1 to this node's count, and add the child's weight value to the accumulator a.

4. Score.Group.Normalize - Normalize this node's score: compute s = s / a.

5. Score.Weight - Assign the node a weighted score equal to the product of its score and its weight.

The final test score is the normalized score value on the root node of the tree, which is the Benchmark object.

5.3.6.2. The Flat Model

This model is identified by the URI "urn:xccdf:scoring:flat".

Under this model, the set of Rule results is treated as a vector V, as described above. The following algorithm SHALL be used to compute the score.

1. Score.Init - Initialize both the score s and the maximum score m to 0.0.

2. Score.Rules - For each element e in V where e.p is not a member of the set {notapplicable, notchecked, informational, notselected}:

- add the weight of rule e.r to m
- if the value e.p equals 'pass' or 'fixed', add the weight of the rule e.r to s.

Thus, the flat model simply computes the sum of the weights for the Rules that passed as the score, and the sum of the weights of all the applicable Rules as the maximum possible score. This model is simple and easy to compute, but scores between different target systems might not be directly comparable because the maximum score can vary.

5.3.6.3. The Flat Unweighted Model

This model is identified by the URI "urn:xccdf:scoring:flat-unweighted". It is computed in exactly the same way as the flat model, except that all weights are taken to be 1.0.

5.3.6.4. The Absolute Model

This model is identified by the URI "urn:xccdf:scoring:absolute". It gives a score of 1 only when all applicable rules in the benchmark pass. It is computed by applying the Flat Model and returning 1 if s=m, and 0 otherwise.

5.3.7. Multiply-Instantiated Rules

A security auditor applying a security guidance document to a system typically wants to know two things: how well does the system comply, and how can non-compliant items be reconciled (either fixed or determined not to be salient)?

Many XCCDF documents include Rules that apply to system components. For example, a host OS Benchmark would probably contain Rules that apply to all users, and a router Benchmark will contain Rules that apply to all network interfaces. When the system holds many of such components, it is not adequate for a tool to inform the administrator or auditor that a Rule failed; it SHOULD report exactly which components failed the Rule.

A processing engine that performs a Benchmark compliance test MAY deliver zero or more pass/fail triples, as described above. In the most common case, each compliance test Rule will yield one result

element. In a case where a Rule was applied multiple times to multiple components of the system under test, a single Rule MAY yield multiple result elements. If each of multiple relevant components passes the Rule, the processing engine MAY deliver a single result element with an instance set I=null. For the purposes of scoring, a Rule SHALL contribute to the positive score only if all instances of that Rule have a test result of 'pass'. If any component of the target system fails a Rule, then the entire Rule SHALL be considered to have failed. This is sometimes called "strict scoring".

6. XML Representation

This section defines a concrete representation of the XCCDF data model in XML, using both core XML syntax and XML Namespaces.

6.1. XML Document General Considerations

The basic document format consists of a root "Benchmark" element, representing a Benchmark object. Its child elements are the contents of the Benchmark object, as described in Section 5.2.

All the XCCDF elements in the document SHALL belong to the XCCDF namespace, including the root element. The namespace URI corresponding to this version of the specification is "http://checklists.nist.gov/xccdf/1.1". The namespace of the root Benchmark element SHALL identify the XCCDF version for a document. Applications that process XCCDF MAY use the namespace URI to decide whether or not they can process a given document. If a namespace prefix is used, the suggested prefix string is "cdf".

XCCDF attributes are not namespace qualified. All attributes begin with a lowercase letter, except the "Id" attribute (for compatibility with XML Digital Signatures [6]).

The example below illustrates the outermost structure of an XCCDF XML document.

```
<?xml version="1.0" ?>
<cdf:Benchmark id="example1" xml:lang="en" Id="toSign"
    xmlns:htm="http://www.w3.org/1999/xhtml"
    xmlns:cdf="http://checklists.nist.gov/xccdf/1.1"
    xmlns:cpe="http://cpe.mitre.org/dictionary/2.0"/>
  <cdf:status date="2004-10-12">draft</cdf:status>
  <cdf:title>Example Benchmark File</cdf:title>
```

```
    <cdf:description>A <htm:b>Small</htm:b> Example
    </cdf:description>
    <cdf:platform idref="cpe:/o:cisco:ios:12.0"/>
    <cdf:version>0.2</cdf:version>
    <cdf:reference href="http://www.ietf.org/rfc/rfc822.txt">
        Standard for the Format of ARPA Internet Text Messages
    </cdf:reference>
  </cdf:Benchmark>
```

Validation is strongly RECOMMENDED but NOT REQUIRED for tools that process XCCDF documents. The XML Schema attribute 'schemaLocation' MAY be used to refer to the XCCDF Schema.

Properties of XCCDF objects marked as type 'text' in Section 5.2 MAY contain embedded formatting, presentation, and hyperlink structure. XHTML Basic tags MUST be used to express the formatting, presentation, and hyperlink structure within XCCDF documents. In particular, the core modules noted in the XHMTL Basic Recommendation [3] are permitted in XCCDF documents, plus the Image module and the Presentation module. How an XCCDF processing tool handles embedded XHTML content in XCCDF text properties is implementation-dependent, but at the least every tool MUST be able to process XCCDF files even when embedded XHTML elements are present. Tools that perform document generation processing SHOULD attempt to preserve the formatting semantics implied by the Text and List modules, support the link semantics implied by the Hypertext module, and incorporate the images referenced via the Image module.

6.2. XML Element Dictionary

This subsection describes each of the elements and attributes of the XCCDF XML specification. Each description includes the parent elements feasible for that element, as well as the child elements it might normally contain. Most elements are in the XCCDF namespace, which for version 1.1.4 is "http://checklists.nist.gov/xccdf/1.1".

Many of the elements listed below are described as containing formatted text (type 'text' in Section 5.2). These elements MAY contain Value substitutions and formatting expressed as described in Section 6.3.

XML is case-sensitive. The XML syntax for XCCDF follows a common convention for representing object-oriented data models in XML: elements that correspond directly to object classes in the data model have names with initial caps. REQUIRED attributes and elements

are denoted by a '+'. Child elements are listed in the order in which they MUST appear. Elements which are not part of the XCCDF namespace are denoted by a '*'.

6.2.1. <Benchmark>

This is the root element of the XCCDF document; it MUST appear exactly once. It encloses the entire Benchmark, and contains both descriptive information and Benchmark structural information. The id attribute MUST be a unique identifier.

- Content: elements
- Cardinality: 1
- Parent Element: none
- Attributes: id+, resolved, style, style-href, xml:lang, Id (note: "Id" is needed only for digital signature security)
- Child Elements: status, title+, description, notice, front-matter, rear-matter, reference, platform-specification*, platform, version, metadata, Profile, Value, Group, Rule, signature

Note that the order of Group and Rule child elements might matter for the appearance of a generated document. Group and Rule children MAY be freely intermingled, but they MUST appear after any Value children. All the other children MUST appear in the order shown, and multiple instances of a child element MUST be adjacent.

6.2.2. <Group>

A Group element contains descriptive information about a portion of a Benchmark, as well as Rules, Values, and other Groups. A Group MUST have a unique id attribute to be referenced from other XCCDF documents or extended by other Groups. The 'extends' attribute, if present, MUST have a value equal to the id attribute of another Group. The 'cluster-id' attribute is an id; it designates membership in a cluster of Items, which are used for controlling Items via Profiles. The 'hidden' and 'allowChanges' attributes are of boolean type and default to false. The weight attribute is a positive real number.

- Content: Elements
- Cardinality: 0-n

- Parent Elements: Benchmark, Group
- Attributes: id+, cluster-id, extends, hidden, prohibitChanges, selected+, weight, Id
- Child Elements: status, version, title, description, warning, question, reference, rationale, platform, requires, conflicts, Value, Group, Rule

All child elements are OPTIONAL, but every group SHOULD have a title, as this will help human editors and readers understand the purpose of the Group. Group and Rule children MAY be freely intermingled. All the other children MUST appear in the order shown, and multiple instances of a child element MUST be adjacent.

The extends attribute allows a Benchmark author to define a group as an extension of another group. The example XML fragment below shows an example of an extended and extending Group.

```
<cdf:Group id="basegrp" selected="0" hidden="1">
  <cdf:title>Example Base Group</cdf:title>
  <cdf:reference>Consult the vendor documentation.
  </cdf:reference>
</cdf:Group>
<cdf:Group extends="basegrp" id="fileperm" selected="1">
  <cdf:title>File Permissions</cdf:title>
  <cdf:description>Rules related to file access control and
      user permissions.</cdf:description>
  <cdf:question>Include checks for file access controls?
  </cdf:question>
  <cdf:reference
      href="http://www.vendor.com/docs/perms.html">
      Administration manual, permissions settings reference
  </cdf:reference>
  . . .
</cdf:Group>
```

An XCCDF Group MAY only extend a Group that is within its visible scope. The visible scope includes sibling elements, siblings of ancestor elements, and the visible scope of any Group that an ancestor Group extended.

Circular dependencies of extension SHALL NOT be defined.

6.2.3. <Rule>

A Rule element defines a single Item to be checked as part of a Benchmark, or an extendable base definition for such Items. A Rule MUST have a unique id attribute, and this id SHALL be used when the Rule is used for extension, referenced from Profiles, or referenced from other XCCDF documents.

The 'extends' attribute, if present, MUST have a value equal to the id attribute of another Rule. The 'weight' attribute MUST be a positive real number. Rules SHALL NOT be nested.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Benchmark, Group
- Attributes: id+, cluster-id, extends, hidden, multiple, prohibitChanges, role, selected, severity, weight, Id
- Child Elements: status, version, title, description, warning, question, reference, rationale, platform, requires, conflicts, ident, profile-note, fixtext, fix, complex-check, check

The check child element of a Rule is the vital piece that specifies how to check compliance with a security practice or guideline. See the description of the check element below for more information. Example 3 shows a very simple Rule element.

```
<cdf:Rule id="pwd-perm" selected="1" weight="6.5" severity="high">
  <cdf:title>Password File Permission</cdf:title>
  <cdf:description>Check the access control on the password
      file.  Normal users should not be able to write to it.
  </cdf:description>
  <cdf:requires idref="passwd-exists"/>
  <cdf:fixtext>
      Set permissions on the passwd file to owner-write,
      world-read
  </cdf:fixtext>
  <cdf:fix strategy="restrict" reboot="0" disruption="low">
      chmod 644 /etc/passwd
  </cdf:fix>
  <cdf:check system="http://www.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="ovaldefs.xml"
```

```
          name="OVAL123"/>
    </cdf:check>
  </cdf:Rule>
```

An XCCDF Rule MAY only extend a Rule that is within its visible scope. The visible scope includes sibling Rules, Rules that are siblings of ancestor Groups, and the visible scope of any Group that an ancestor Group extended.

Circular dependencies of extension SHALL NOT be defined.

6.2.4. <Value>

A Value element represents a named parameter whose title or value MAY be substituted into other strings in the Benchmark (depending on the form of processing to which the Benchmark is being subjected), or it MAY represent a basis for the definition of such parameters via extension. A Value object MUST have a unique id attribute to be referenced for substitution or extension or for inclusion in another Benchmark.

A Value object MAY appear as a child of the Benchmark, or as a child of a Group. Value objects SHALL NOT be nested. The value and default child elements MUST appear first.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Benchmark, Group
- Attributes: id+, cluster-id, extends, hidden, prohibitChanges, operator, type, interactive, interfaceHint, Id
- Child Elements: status, version, title, description, warning, question, reference, value+, default, match, lower-bound, upper-bound, choices, source

The type attribute is OPTIONAL, but if it appears it MUST be one of 'number', 'string', or 'boolean'. A tool performing tailoring processing MAY use this type name to perform user input validation. The example below shows a very simple Value object.

```
<cdf:Value id="web-server-port" type="number" operator="equals">
  <cdf:title>Web Server Port</cdf:title>
  <cdf:description>TCP port on which the server listens
```

```
    </cdf:description>
    <cdf:value>12080</cdf:value>
    <cdf:default>80</cdf:default>
    <cdf:lower-bound>0</cdf:lower-bound>
    <cdf:upper-bound>65535</cdf:upper-bound>
  </cdf:Value>
```

(Note that the match element SHALL apply only for validation during XCCDF tailoring, while the operator attribute SHALL apply only for rule checking. People often confuse these.)

6.2.5. <Profile>

A Profile element encapsulates a tailoring of the Benchmark. It consists of an id, descriptive text properties, and zero or more selectors that refer to Group, Rule, and Value objects in the Benchmark. There are three selector elements: select, set-value, and refine-value.

Profile elements MAY only appear as direct children of the Benchmark element. A Profile MAY be defined as extending another Profile, using the ’extends’ attribute.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: abstract, id+, extends, prohibitChanges, Id, note-tag
- Child Elements: status, version, title+, description, reference, platform, select, set-value, refine-value, refine-rule

Profiles are designed to support encapsulation of a set of tailorings. A Profile implicitly includes all the Groups and Rules in the Benchmark, and the select element children of the Profile affect which Groups and Rules are selected for processing when the Profile is in effect. The example below shows a very simple Profile.

```
<cdf:Profile id="strict" prohibitChanges="1"
    extends="lenient" note-tag="strict-tag">
  <cdf:title>Strict Security Settings</cdf:title>
  <cdf:description>
      Strict lockdown rules and values, for hosts deployed to
```

```
        high-risk environments.
    </cdf:description>
    <cdf:set-value idref="password-len">10</cdf:set-value>
    <cdf:refine-value idref="session-timeout" selector="quick"/>
    <cdf:refine-rule idref="session-auth-rule" selector="harsh"/>
    <cdf:select idref="password-len-rule" selected="1"/>
    <cdf:select idref="audit-cluster" selected="1"/>
    <cdf:select idref="telnet-disabled-rule" selected="1"/>
    <cdf:select idref="telnet-settings-cluster" selected="0"/>
  </cdf:Value>
```

6.2.6. <TestResult>

The TestResult object encapsulates the result of applying a Benchmark to one target system. The TestResult element SHOULD normally appear as the child of the Benchmark element, although it MAY also appear as the top-level element of a file.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: id+, start-time, end-time+, Id
- Child Elements: title, remark, organization, identity, profile, set-value, target, target-address, target-facts, rule-result, score

The id attribute is a REQUIRED unique-id for a test result. The start-time and end-time attributes MUST have the format of a timestamp; the end-time attribute is REQUIRED, and gives the time that the application of the Benchmark completed.

The example below shows a TestResult object with a few rule-result children.

```
<cdf:TestResult id="ios-test5"
    end-time="2007-09-25T7:45:02-04:00"
    xmlns:cdf="http://checklists.nist.gov/xccdf/1.1">
  <cdf:benchmark href="ios-sample-12.4.xccdf.xml"/>
  <cdf:title>Sample Results Block</cdf:title>
  <cdf:remark>Test run by Bob on Sept 25, 2007</cdf:remark>
```

```
    <cdf:organization>Department of Commerce</cdf:organization>
    <cdf:organization>National Institute of Standards and
  Technology</cdf:organization>
    <cdf:identity authenticated="1" privileged="1">admin_bob
    </cdf:identity>
    <cdf:target>lower.test.net</cdf:target>
    <cdf:target-address>192.168.248.1</cdf:target-address>
    <cdf:target-address>2001:8::1</cdf:target-address>
    <cdf:target-facts>
      <cdf:fact type="string"
          name="urn:xccdf:fact:ethernet:MAC">
          02:50:e6:c0:14:39
      </cdf:fact>
      <cdf:fact name="urn:xccdf:fact:OS:IOS">1</cdf:fact>
    </cdf:target-facts>
    <cdf:set-value idref="exec-timeout-time">10</cdf:set-value>
    <cdf:rule-result idref ="ios12-no-finger-service"
        time="2007-09-25T13:45:00-04:00">
      <cdf:result>pass</cdf:result>
    </cdf:rule-result>
    <cdf:rule-result idref ="req-exec-timeout"
        time="2007-09-25T13:45:06-04:00">
      <cdf:result>fail</cdf:result>
      <cdf:instance>console</cdf:instance>
      <cdf:fix>
          line console
          exec-timeout 10 0
      </cdf:fix>
    </cdf:rule-result>
    <cdf:score>67.5</cdf:score>
    <cdf:score system="urn:xccdf:scoring:absolute">0
    </cdf:score>
  </cdf:TestResult>
```

6.2.7. Other Elements and Attributes

This subsection describes other elements and attributes, which are listed in alphabetical order.

6.2.7.1. <benchmark>

This simple element SHALL only appear as the child of a TestResult. It indicates the Benchmark for which the TestResult records results. It has one attribute, which gives the URI of the Benchmark XCCDF document. It MUST be an empty element.

- Content: none
- Cardinality: 0-1
- Parent Elements: TestResult
- Attributes: href+
- Child Elements: none

The Benchmark element SHOULD be used only in a standalone TestResult file (an XCCDF document file whose root element is TestResult).

6.2.7.2. <check>

This element holds a specification for how to check compliance with a Rule. It MAY appear as a child of a Rule element, or in somewhat abbreviated form as a child of a rule-result element inside a TestResult object.

The child elements of the check element specify the values to pass to a checking engine, and the logic for the checking engine to apply. The logic MAY be embedded directly as inline text or XML data, or MAY be a reference to an element of an external file indicated by a URI. If the compliance checking system uses XML namespaces, then the system attribute for the system SHOULD be its namespace. The default or nominal content for a check element is a compliance test expressed as an OVAL Definition or a reference to an OVAL Definition, with the system attribute set to the OVAL namespace.

The check element MAY also be used as part of a TestResult rule-result element; in that case it holds or refers to detailed output from the checking engine.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Rule, rule-result

- Attributes: id, selector, system+
- Child Elements: check-import, check-export, check-content-ref, check-content

A check element MAY have a selector attribute, which MAY be referenced from a Benchmark Profile as a means of refining the application of the Rule. When Profiles are not used, then all check elements with non-empty selectors SHALL be ignored.

Several check elements MAY appear as children of the same Rule element. Sibling check elements MUST have different values for the combination of their selector and system attributes, and different values for their id attribute (if any). A tool processing the Benchmark for compliance checking MUST pick at most one check or complex-check element to process for each Rule.

The check element SHALL contain zero or more check-import elements, followed by zero or more check-export elements, followed by zero or more check-content-ref elements, followed by at most one check-content element. If two or more check-content-ref elements appear, then they represent alternative locations from which a tool may obtain the check content. Tools SHOULD process the alternatives in order, and use the first one found. If both check-content-ref elements and check-content elements appear, tools SHOULD use the check-content only if all references are inaccessible.

When a check element is a child of a Rule object, check-import and check-export elements MUST be empty. When a check element is a child rule-result object, check-import elements SHALL contain the value retrieved from the checking system.

6.2.7.3. <check-import>

This element identifies a value to be retrieved from the checking system during testing of a target system. The value-id attribute is merely a locally unique id. It MUST match the id attribute of a Value object in the Benchmark.

- Content: string
- Cardinality: 0-n
- Parent Elements: check
- Attributes: import-name

- Child Elements: none

When a check-import element appears in the context of a Rule object, it MUST be empty. When it appears in the context of a rule-result, its content SHALL be the value retrieved from the checking system.

6.2.7.4. <check-export>

This specifies a mapping from an XCCDF Value object to a checking system variable. The value-id attribute MUST match the id attribute of a Value object in the Benchmark.

- Content: none
- Cardinality: 0-n
- Parent Elements: check
- Attributes: value-id, export-name
- Child Elements: none

6.2.7.5. <check-content>

This element holds the actual code of a Benchmark compliance check, in the language or system specified by the check element’s system attribute. At least one of check-content or check-content-ref MUST appear in each check element. The body of this element MAY be any XML, but SHALL NOT contain any XCCDF elements. XCCDF tools are not required to process this element; typically it will be passed to a checking system or engine.

- Content: any non-XCCDF*
- Cardinality: 0-1
- Parent Elements: check
- Attributes: none
- Child Elements: special*

6.2.7.6. <check-content-ref>

This element points to a Benchmark compliance check, in the language or system specified by the check element’s system attribute. At least one of check-content or check-content-ref MUST appear in each

check element. The 'href' attribute identifies the document, and the OPTIONAL name attribute MAY be used to refer to a particular part, element, or component of the document.

- Content: none
- Cardinality: 0-n
- Parent Elements: check
- Attributes: href+, name
- Child Elements: none

6.2.7.7. <choices>

The choices element MAY be a child of a Value, and it enumerates one or more legal values for the Value. If the boolean 'mustMatch' attribute is true, then the list represents all the legal values; if mustMatch is absent or false, then the list represents suggested values, but other values might also be legal (subject to the parent Value's upper-bound, lower-bound, or match attributes). The choices element MAY have a selector attribute that is used for tailoring via a Profile. The list given by this element is intended for use during tailoring and document generation; it has no role in Benchmark compliance checking.

- Content: elements
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: mustMatch, selector
- Child Elements: choice+

6.2.7.8. <choice>

This string element is used to hold a possible legal value for a Value object. It MUST appear as the child of a choices element.

- Content: string
- Cardinality: 1-n
- Parent Elements: choices

- Attributes: none
- Child Elements: none

If a tool presents the choice values from a choices element to a user, they SHOULD be presented in the order in which they appear.

6.2.7.9. <complex-check>

This element SHALL only appear as a child of a Rule. It contains a boolean expression composed of operators (and, or, not) and individual checks.

- Content: elements
- Cardinality: 0-1
- Parent Elements: Rule
- Attributes: operator+, negate
- Child Elements: complex-check, check

Truth tables for boolean operation in complex checks are given below; all the abbreviations in the truth tables come from the description of the 'result' element in the TestResult object (see Section 6.2.6).

With an "AND" operator, the complex-check SHALL evaluatee to Pass only if all of its enclosed terms (checks and complex-checks) evaluate to Pass. For purposes of evaluation, Pass (P) and Fixed (X) are considered equivalent. The truth table for "AND" is given below.

AND	P	F	U	E	N
P	P	F	U	E	P
F	F	F	F	F	F
U	U	F	U	U	U
E	E	F	U	E	E
N	P	F	U	E	N

The 'OR' operator SHALL evaluate to Pass if any of its enclosed terms evaluate to Pass. The truth table for "OR" is given below.

OR	P	F	U	E	N
P	P	P	P	P	P

```
F     P  F  U  E  F
U     P  U  U  U  U
E     P  E  U  E  E
N     P  F  U  E  N
```

If the negate attribute is set to true, then the result of the complex-check MUST be complemented (inverted). The full truth table for negation is given below.

```
      P  F  U  E  N
not   F  P  U  E  N
```

The example below shows a complex-check with several components.

```
<cdf:complex-check operator="OR">
  <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="xpDefs.xml" name="XP-P1"/>
  </cdf:check>
  <cdf:complex-check operator="AND" negate="1">
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
      <cdf:check-content-ref href="xpDefs.xml" name="XP-CX"/>
    </cdf:check>
    <cdf:check
system="http://www.cisecurity.org/xccdf/interactive/1.0">
      <cdf:check-content-ref href="xpInter.xml" name="XP-6"/>
    </cdf:check>
  </cdf:complex-check>
</cdf:complex-check>
```

6.2.7.10. <conflicts>

The conflicts element MAY be a child of any Group or Rule, and it specifies a list of the id properties of other Group or Rule item whose selection conflicts with this one. Each conflicts element specifies a single conflicting Item using its 'idref' attribute; if the semantics of the Benchmark need multiple conflicts, then multiple conflicts elements MAY appear. A conflicts element MUST be empty.

- Content: none
- Cardinality: 0-n

- Parent Elements: Group, Rule
- Attributes: idref+
- Child Elements: none

6.2.7.11. <cpe-list>

This element holds names and descriptions for one or more platforms, using the XML schema defined for the Common Platform Enumeration (CPE) 1.0. CPE Names are URIs, and MAY be used for all platform identification in an XCCDF document. This element is deprecated, and appears in the XCCDF 1.1.4 specification only for compatibility with earlier versions.

- Content: elements (from the CPE 1.0 dictionary namespace)
- Cardinality: 0-1
- Parent Elements: Benchmark
- Attributes: none
- Child Elements: cpe-item*

6.2.7.12. <default>

This string element is used to hold the default or reset value of a Value object. It SHALL only appear as a child of a Value element. This element MAY have a selector attribute, which MAY be used to designate different defaults for different Benchmark Profiles.

- Content: string
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: selector
- Child Elements: none

6.2.7.13. <description>

This element provides the descriptive text for a Benchmark, Rule, Group, or Value. Multiple description elements MAY appear with

different values for their xml:lang attribute (see also next section).

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Benchmark, Group, Rule, Value, Profile
- Attributes: xml:lang, override
- Child Elements: sub, xhtml elements*

The ’sub’ element MAY appear inside a description, and in many other descriptive text elements. During document generation, each instance of the ’sub’ element SHOULD be replaced by the title of the Item or other object to which it refers. For more information, see Section 5.3.3.

6.2.7.14. <fact>

This element holds a single type-name-value fact about the target of a test. The name is a URI. Pre-defined names start with "urn:xccdf:fact", but tool developers MAY define additional platform-specific and tool-specific facts.

- Content: string
- Cardinality: 0-n
- Parent Elements: target-facts
- Attributes: name+, type
- Child Elements: none

The following types are supported: "number", "string", and "boolean" (the default).

6.2.7.15. <fix>

This element MAY appear as the child of a Rule element, or a rule-result element. When it appears as a child of a Rule element, it contains string data for a command, script, or procedure that should bring the target into compliance with the Rule. It SHALL NOT contain XHTML formatting. The fix element MAY contain XCCDF Value

substitutions specified with the sub element, or instance name substitution specified with an instance element.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Rule, rule-result
- Attributes: id, complexity, disruption, platform, reboot, strategy, system
- Child Elements: instance, sub

The fix element supports several attributes that the Rule author MAY use to provide additional information about the remediation that the fix element contains. The attributes and their permissible values are listed below.

id - A local id for the fix, which allows fixtext elements to refer to it. It is OPTIONAL for ids to be unique; several fix elements might have the same id but different values for the other attributes.

complexity - A keyword that indicates the complexity or difficulty of applying the fix to the target. Allowed values:

- unknown - default, complexity not defined
- low - the fix is very simple to apply
- medium - fix is moderately difficult or complex
- high - the fix is very complex to apply

disruption - A keyword that designates the potential for disruption or degradation of target operation. Allowed values:

- unknown - default, disruption not defined
- low - little or no disruption expected
- medium - potential for minor or short-lived disruption
- high - potential for serious disruption

platform - A platform identifier; this SHOULD appear on a fix when the content applies to only one platform out of several to which the Rule could apply.

reboot - boolean - if remediation will require a reboot or hard reset of the target ('1' means reboot required)

strategy - A keyword that designates the approach or method that the fix uses. Allowed values:

- unknown - default, strategy not defined
- configure - adjust target configuration/settings
- patch - apply a patch, hotfix, update, etc.
- disable - turn off or uninstall a target component
- enable - turn on or install a target component
- restrict - adjust permissions, access rights, filters, or other access restrictions
- policy - remediation requires out-of-band adjustments to policies or procedures
- combination - strategy is a combination or two or more approaches

system - A URI that identifies the scheme, language, or engine for which the fix is written. Several general URIs are defined, but platform-specific URIs MAY be expected. (For a list of pre-defined fix system URIs, see Appendix A.)

The platform attribute defines the platform for which the fix is intended, if its parent Rule applied to multiple platforms. The value of the platform attribute SHOULD be one of the platform strings defined for the Benchmark. If the fix's platform attribute is not given, then the fix SHALL apply to all platforms to which its enclosing Rule applies.

As a special case, fix elements MAY also appear as children of a rule-result element in a TestResult. In this case, the fix element SHOULD NOT have any child elements, and its content should be a simple string. When a fix element is the child of rule-result, it is assumed to have been 'instantiated' by the testing tool, and any substitutions or platform selection already made.

6.2.7.16. <fixtext>

This element, which SHALL only appear as a child of a Rule element, provides text that explains how to bring a target system into compliance with the Rule. Multiple instances MAY appear in a Rule, with different attribute values.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Rule
- Attributes: xml:lang, fixref, disruption, reboot, strategy, override
- Child Elements: sub, xhtml elements*

The fixtext element and its counterpart, the fix element, are fairly complex. They can accept a number of attributes that describe aspects of the remediation. The xml:lang attribute designates the locale for which the text was written; it is expected that fix elements usually will be locale-independent. The following attributes MAY appear on the fixtext element (for details about most of them, refer to the table under the fix element definition in Section 6.2.7.15).

- fixref - A reference to the id of a fix element
- complexity - A keyword that indicates the difficulty or complexity of applying the described fix to the target
- disruption - A keyword that designates the potential for disruption or degradation of target operation
- reboot - boolean - if the remediation described in the fixtext will require a reboot or reset of the target
- strategy - A keyword that designates the approach or method that the fix uses

The fixtext element MAY contain XHTML elements, to aid in formatting.

6.2.7.17. <front-matter>

This element contains textual content intended for use during Document Generation processing only; it is introductory matter that SHOULD appear at or near the beginning of the generated document. Multiple instances MAY appear with different xml:lang values.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: xml:lang
- Child Elements: sub, xhtml elements*

6.2.7.18. <ident>

This element contains a string (name) which is a long-term globally meaningful identifier in some naming scheme. The content of the element is the name, and the system attribute contains a URI which designates the organization or scheme that assigned the name (see Appendix A for assigned URIs).

- Content: string
- Cardinality: 0-n
- Parent Elements: Rule, rule-result
- Attributes: system+
- Child Elements: none

See Section 6.2.7.21 for an example of this element.

6.2.7.19. <identity>

This element SHALL appear only as a child of a TestResult. It provides up to three pieces of information about the system identity or user employed during application of the Benchmark: whether the identity was authenticated, whether the identity was granted administrative or other special privileges, and the name of the identity.

- Content: string

- Cardinality: 0-1
- Parent Elements: TestResult
- Attributes: authenticated+, privileged+
- Child Elements: none

The attributes are both REQUIRED, and both boolean. The string content of the element is the identity name, and MAY be omitted.

6.2.7.20. <impact-metric>

This element contains a string representation of the potential impact of failure to conform to a Rule. The content MUST be a CVSS base vector, expressed using the format defined in the CVSS 2.0 specification [7].

- Content: string
- Cardinality: 0-1
- Parent Elements: Rule
- Attributes: none
- Child Elements: none

The example below shows how the ident and impact-metric elements can be used to associate a Rule with a Common Configuration Enumeration identifier and a CVSS score.

```
<cdf:Rule id="debug.exePermissions" selected="1"
    weight="10.0">
  <cdf:title>debug.exe Permissions</cdf:title>
  <cdf:description>
      Failure to properly configure ACL file and directory
      Permissions allows the possibility of unauthorized and
      Anonymous modifications to the operating system and
      installed applications.
  </cdf:description>
  <cdf:platform idref="cpe:/o:microsoft:windows-nt:xp"/>
  <cdf:ident system="http://cce.mitre.org/">CCE-201
  </cdf:ident>
  <cdf:impact-metric>AV:L/AC:L/Au:S/C:P/I:P/A:N
```

```
    </cdf:impact-metric>
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval-
  definitions-5">
      <check-content-ref href="winxppro.xml"
          name="oval:gov.nist.1:def:133"
    </cdf:check>
  </cdf:Rule>
```

6.2.7.21. <instance>

The instance element MAY appear in two situations. First, it MAY appear as part of a TestResult, as a child of a rule-result element; in that situation it contains the name of a target component to which a Rule was applied, in the case of multiply-instantiated rules.

- Content: string
- Cardinality: 0-n
- Parent Elements: rule-result
- Attributes: context, parentContext
- Child Elements: none

If the context attribute is omitted, the value of the context defaults to "undefined". At most one instance child of a rule-result MAY have a context of "undefined".

Second, the instance element MAY appear as part of a Rule, as a child of the fix element. In that situation it represents a place in the fix text where the name of a target component SHOULD be substituted, in the case of multiply-instantiated rules.

- Content: none
- Cardinality: 0-n
- Parent Elements: fix
- Attributes: context
- Child Elements: none

If the context attribute is omitted, the value of the context defaults to "undefined".

6.2.7.22. <lower-bound>

This element MAY appear one or more times as a child of a Value element. It is used to constrain value input during tailoring, when the Value's type is "number". It contains a number; values supplied by the user for tailoring the Benchmark MUST be no less than this number. This element MAY have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one lower-bound element appears as the child of a Value, at most one of them MAY omit the selector attribute.

- Content: number
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: selector
- Child Elements: none

6.2.7.23. <match>

This element MAY appear one or more times as a child of a Value element. It is used to constrain value input during tailoring. It contains a regular expression that a user's input for the value MUST match. This element MAY have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one match element appears as the child of a Value, at most one of them MAY omit the selector attribute.

- Content: string
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: selector
- Child Elements: none

6.2.7.24. <message>

This element is OPTIONAL, but MAY appear one or more times as a child of a rule-result element inside a TestResult object. It holds a single informational or error message that was returned by the checking engine.

- Content: string
- Cardinality: 0-n
- Parent Elements: rule-result
- Attributes: severity+
- Child Elements: none

The severity attribute denotes the seriousness or conditions of the message. In XCCDF 1.1.4 there are three message severity values: "error", "warning", and "info". These elements SHALL NOT affect scoring; they are present merely to convey diagnostic information from the checking engine. XCCDF tools that deal with TestResult data might choose to log these messages or display them to the user.

6.2.7.25. <metadata>

The metadata element is OPTIONAL, but MAY appear one or more times as a child of the Benchmark element. It contains document metadata expressed in XML. The default format for Benchmark document metadata is the Dublin Core Metadata Initiative (DCMI) Simple DC Element specification, as described in [12] and [13]. An example of the default format is shown in the example below. Tools, especially document generation tools, SHOULD be prepared to process Dublin Core metadata in this element.

- Content: element
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: none
- Child Elements: non-XCCDF* (Dublin Core elements recommended)

```
<cdf:metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
```

```
    <dc:title>Security Benchmark for Ethernet Hubs</dc:title>
    <dc:creator>James Smith</dc:creator>
    <dc:publisher>Center for Internet Security</dc:publisher>
    <dc:subject>network security for layer 2 devices
    </dc:subject>
  </cdf:metadata>
```

Another suitable metadata format for XCCDF Benchmarks is the XML description format mandated by NIST for its National Checklist Program [10].

6.2.7.26. <model>

This element contains a specification for a suggested scoring model. This element SHALL only appear as a child of a Benchmark, and has one REQUIRED attribute: the URI of the scoring model. Some models might need additional parameters; to support such a model, one or more 'param' elements MAY appear as children of the 'model' element.

- Content: element
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: system+
- Child Elements: param

This specification defines the following four scoring model URIs; compliance checking tool developers MAY define additional models. For more information see Section 5.3.

- urn:xccdf:scoring:default - this is the default weighted aggregated model. All tools MUST support this model, and it is the default for Benchmarks that do not include any other model specifications.
- urn:xccdf:scoring:flat - this model computes the sum of the weights of rules that pass. All tools SHOULD support this model.
- urn:xccdf:scoring:flat-unweighted - this simple model simply computes the number of rules that passed. It does not use weights. All tools SHOULD support this model.

- urn:xccdf:scoring:absolute - this extremely simple model gives a score of 1 if all applicable rules passed, and 0 otherwise.

6.2.7.27. <new-result>

An override rule result status. This element appears in an override element, inside a rule-result element, in a TestResult object. Its content MUST be one of the result status values listed in Section 5.2.

- Content: string
- Cardinality: 1
- Parent Elements: override
- Attributes: none
- Child Elements: none

6.2.7.28. <notice>

This string element SHALL only appear as the child of a Benchmark element, and supplies legal notice or copyright text about the Benchmark document. If specified, the id attribute MUST be a unique identifier.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: id
- Child Elements: xhtml elements*

The notice element MAY contain XHTML markup to give it internal structure and formatting.

6.2.7.29. <old-result>

This element holds an overridden rule result status. This element appears in an override element, inside a rule-result element, in a TestResult object. Its content MUST be one of the result status values listed in Section 5.2.

- Content: string
- Cardinality: 1
- Parent Elements: override
- Attributes: none
- Child Elements: none

6.2.7.30. <organization>

This element SHALL appear only as a child of a TestResult. It contains the name of the organization, department, or other entity responsible for the results.

- Content: string
- Cardinality: 0-n
- Parent Elements: TestResult
- Attributes: none
- Child Elements: none

When multiple organization elements appear in a TestResult to indicate multiple organization names in a hierarchical organization, the highest-level organization SHOULD appear first (e.g., "U.S. Government") followed by subordinate organizations (e.g., "Department of Defense").

6.2.7.31. <override>

This element SHALL appear only as a child of a rule-result, and represents a human override of a Benchmark rule check result. It consists of five parts: a timestamp, the name of the human authority for the override, the old and new result status values, and remark text.

- Content: elements
- Cardinality: 0-n
- Parent Elements: rule-result
- Attributes: time+, authority

- Child Elements: old-result, new-result, remark

The example below shows how an override block would appear in a rule-result.

```
<cdf:rule-result idref="rule76"
    time="2005-04-26T14:38:19Z" severity="low">
  <cdf:result>pass</cdf:result>
  <cdf:override time="2005-04-26T15:03:20Z"
      authority="Bob Smith">
    <cdf:old-result>fail</cdf:old-result>
    <cdf:new-result>pass</cdf:new-result>
    <cdf:remark>
        Manual inspection showed this rule be satisfied.
        The relevant registry key was protected per policy,
        but with a more restrictive ACL than the benchmark
        was designed to check. The rule result has been
        overridden to 'pass'.
    </cdf:remark>
  </cdf:override>
  <cdf:instance context="registry">
        HKLM\SOFTWARE\Policies
  </cdf:instance>
  <cdf:check system="http://www.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="oval-out.xml"
        name="OVAL123"/>
  </cdf:check>
</cdf:rule-result>
```

Note: if an override is added to a rule-result, it will break any digital signature applied to the enclosing TestResult object.

6.2.7.32. <param>

This element SHALL appear only as a child of a model element. It supplies a parameter that the compliance checking tool will need when computing the score using that model. Parameters are NOT REQUIRED by any of the scoring models defined in the XCCDF specification, but proprietary models MAY require parameters.

- Content: string

- Cardinality: 0-n

- Parent Elements: model
- Attributes: name+
- Child Elements: none

Param elements with equal values for the name attribute SHALL NOT appear as children of the same model element.

6.2.7.33. <plain-text>

This element holds a re-usable chunk of text for a Benchmark. It MAY be used anywhere that XHTML content or the XCCDF sub element may be used. Each plain-text definition MUST have a unique id.

- Content: string
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: id+
- Child Elements: none

Note that plain-text definitions SHALL only appear as children of Benchmark. The id on a plain-text definition MUST NOT be equal to any of the ids on Value, Group, or Rule objects.

6.2.7.34. <platform>

The ’platform’ element specifies a target platform for the Benchmark or a particular Item, Profile, or TestResult. This element has no content; the platform identifier appears in the ’idref’ attribute. Multiple ’platform’ elements MAY appear as children of a Benchmark, Item, or Profile, if the Benchmark or Item is suitable for multiple kinds of target systems.

- Content: none
- Cardinality: 0-n
- Parent Elements: Benchmark, Group, Rule, Value, TestResult
- Attributes: idref+, override
- Child Elements: none

The platform element is OPTIONAL. It indicates a platform with a CPE Name (URI) or a CPE platform specification identifier. The URI or identifier appears in the idref attribute.

The override attribute SHALL NOT appear when the platform element is a child of a TestResult.

6.2.7.35. <platform-specification>

The platform-specification element defines a list of identifiers for compound platform definitions written in the CPE 2.2 Language [5]. The identifiers defined in the platform-specification element MAY be used anywhere in the Benchmark in place of a CPE Name.

- Content: elements (from the CPE 2.2 language namespace)
- Cardinality: 0-1
- Parent Elements: Benchmark
- Attributes: none
- Child Elements: special*

6.2.7.36. <platform-definitions>, <Platform-Specification>

Each of these elements contains information about platforms to which the Benchmark may apply. The <platform-definitions> element contains a set of platform component and platform definitions using the CIS platform schema; it is permitted in XCCDF 1.1.4 for compatibility with 1.0. Both of these elements are deprecated, and appear in this specification only for backward compatibility. Common Platform Enumeration (CPE) Names SHOULD be used instead, see Section 6.2.7.11.

- Content: elements
- Cardinality: 0-1
- Parent Elements: Benchmark
- Attributes: none
- Child Elements: special*

6.2.7.37. <profile>

This element specifies the Benchmark Profile used in applying a Benchmark; it SHALL appear only as a child of a TestResult element.

- Content: none
- Cardinality: 0-1
- Parent Elements: TestResult
- Attributes: idref+
- Child Elements: none

6.2.7.38. <profile-note>

This element holds descriptive text about how one or more Profiles affect a Rule. It SHALL appear only as a child of the Rule element.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Rule
- Attributes: tag+, xml:lang
- Child Elements: sub, xhtml elements*

The mandatory ’tag’ attribute holds an identifier; Profiles MAY refer to this identifier using the ’note-tag’ attribute.

6.2.7.39. <question>

This element specifies an interrogatory string with which to prompt the user during tailoring. It MAY also be included into a generated document. Note that this element SHALL NOT contain any XCCDF child elements or XHTML formatting elements. Multiple instances MAY appear with different xml:lang attributes.

- Content: string
- Cardinality: 0-n
- Parent Elements: Group, Rule, Value

- Attributes: xml:lang, override
- Child Elements: none

For Rule and Group objects, the question text SHOULD be a simple binary (yes/no) question, because tailoring for Rules and Groups is for selection only. For Value objects, the question SHOULD reflect the designed data value needed for tailoring.

6.2.7.40. <rationale>

This element, which MAY appear as a child of a Group or Rule element, provides text that explains why that Group or Rule is important to the security of a target platform.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Group, Rule
- Attributes: xml:lang, override
- Child Elements: sub, xhtml elements*

6.2.7.41. <rear-matter>

This element contains textual content intended for use during Document Generation processing only; it is concluding material that SHOULD appear at or near the end of the generated document. Multiple instances MAY appear with different xml:lang attributes.

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Benchmark
- Attributes: xml:lang
- Child Elements: sub, xhtml elements*

6.2.7.42. <reference>

This element provides supplementary descriptive text for a Benchmark, Rule, Group, or Value. It MAY have a simple string value, or a value consisting of simple Dublin Core elements as described in

[12]. It MAY also have an attribute, 'href', giving a URL for the referenced resource. Multiple reference elements MAY appear; a document generation processing tool MAY concatenate them, or put them into a reference list, and MAY choose to number them.

- Content: string or elements
- Cardinality: 0-n
- Parent Elements: Benchmark, Group, Rule, Value, Profile
- Attributes: xml:lang, href
- Child Elements: none or Dublin Core Elements*

References SHOULD be given as Dublin Core descriptions; a bare string is allowed for simplicity. If a bare string appears, then it is taken to be the string content for a Dublin Core 'title' element. For more information, consult [13].

6.2.7.43. <refine-rule>

This element adjusts the check tailoring selector, weight, severity, and role properties of a Rule. It has five attributes: the id of a Rule (REQUIRED), and adjusted values for check selector, weight, severity, and role (all OPTIONAL).

- Content: none
- Cardinality: 0-n
- Parent Elements: Profile
- Attributes: idref+, role, selector, severity, weight
- Child Elements: none

The 'idref' attribute MUST match the id attribute of a Rule, or a cluster-id of one or more Items in the Benchmark. The 'idref' attribute values of sibling refine-rule element children of a Profile MUST be different. When the 'idref' attribute refers to a Group, or if some of the Items in a cluster are Groups, then only the 'weight' attribute SHALL apply to them.

Selector tags SHALL apply only to the check children of a Rule. If the selector tag specified in a refine-rule element in a Profile does not match any of the selectors specified on any of the check

children of a Rule, then the check child element without a selector attribute MUST be used.

6.2.7.44. <refine-value>

This element specifies the selector tag to be applied when tailoring a Value during use of a particular Profile. It has three attributes: the id of a Value or Item cluster (mandatory), the id of a Value selector tag, and a new setting for the Value operator.

- Content: none
- Cardinality: 0-n
- Parent Elements: Profile
- Attributes: idref+, operator, selector
- Child Elements: none

The 'idref' attribute MUST match the id attribute of a Value, or a cluster-id of one or more Items in the Benchmark. The 'idref' attribute values of sibling refine-value element children of a Profile MUST be different.

The 'operator' attribute of the refine-value element MAY be used to override or replace the operator attribute of a Value. This is useful for refining the semantics of a Value.

Selector tags SHALL apply to the following child elements of Value: choices, default, value, match, lower-bound, and upper-bound. If the selector tag specified in a refine-value element in a Profile does not match any of the selectors specified on any of the Value children, then the child with no selector tag SHALL be used. The example below illustrates how selector tags and the refine-value element work.

```
<cdf:Value id="pw-length" type="number" operator="equals">
  <cdf:title>Minimum password length policy</cdf:title>
  <cdf:value>8</cdf:value>
  <cdf:value selector="high">14</cdf:value>
  <cdf:lower-bound>8</cdf:lower-bound>
  <cdf:lower-bound selector="high">12</cdf:lower-bound>
</cdf:Value>
<cdf:Profile id="enterprise-internet">
```

```
    <cdf:title>Enterprise internet server profile</cdf:title>
    <cdf:refine-value idref="pw-length" selector="high"/>
  </cdf:Profile>
  <cdf:Profile id="home">
    <cdf:title>Home host profile</cdf:title>
  </cdf:Profile>
```

6.2.7.45. <remark>

The remark element MAY appear as the child of a TestResult or override element; it contains a textual remark about the test.

- Content: string
- Cardinality: 0-n (TestResult), 1 (override)
- Parent Elements: TestResult, override, refine-rule, refine-value, select
- Attributes: xml:lang
- Child Elements: none

The remark content SHALL NOT contain any XHTML tags or other structure; it MUST be a plain string.

6.2.7.46. <requires>

The requires element MAY be a child of any Group or Rule, and it specifies the id properties of one or more other Group or Rule which, at least one of which MUST be selected in order for this Item to be selected. In a sense, the requires element is the opposite of the conflicts element. Each requires element specifies a list of one or more required Items by their ids, using the 'idref' attribute. If more than one id is given, then if at least one of the specified Groups or Rules is selected, the requirement is met.

- Content: none
- Cardinality: 0-n
- Parent Elements: Group, Rule
- Attributes: idref+
- Child Elements: none

6.2.7.47. <result>

This simple element holds the verdict of applying a Benchmark Rule to a target or component of a target. It SHALL have only one of nine values: "pass", "fail", "error", "unknown", "notchecked", "notapplicable", "notselected", "fixed" or "informational". For more information see Section 5.2.4.1.

- Content: string
- Cardinality: 1
- Parent Elements: rule-result
- Attributes: None
- Child Elements: None

6.2.7.48. <rule-result>

This element holds the result of applying a Rule from the Benchmark to a target system or component of a target system. It SHALL only appear as the child of a TestResult element.

- Content: elements
- Cardinality: 0-n
- Parent Elements: TestResult
- Attributes: idref+, role, time, severity, version, weight
- Child Elements: result+, override, ident, message, instance, fix, check

The ’idref’ attribute of a rule-result element MUST refer to a Rule element in the Benchmark. The result child element expresses the result (pass, fail, error, etc.) of applying the Rule to the target system. If the Rule is multiply instantiated, the instance elements indicate the particular system component. If present, the override element provides information about a human override of a computed result status value.

6.2.7.49. <score>

This element contains the weighted score for a Benchmark test, as a real number. Scoring models are defined in Section 5.3. This element SHALL only appear as a child of a TestResult element.

- Content: string (non-negative number)
- Cardinality: 1-n
- Parent Elements: TestResult
- Attributes: system, maximum
- Child Elements: none

The system attribute, a URI, identifies the scoring model (see the description of the model element in Section 6.2.7.26 for a list of pre-defined models). If the system attribute does not appear, then the model used was the default model. The maximum attribute, a real number, gives the maximum possible value of the score for this Benchmark test. If the maximum attribute does not appear, then it is taken to have a value of 100.

6.2.7.50. <select>

This element is part of a Profile; it overrides the selected attribute of a Rule or Group. Two attributes MUST be given with this element: the id of a Rule or Group (idref), and a boolean value (selected). If the boolean value is given as true, then the Rule or Group SHALL be selected for this Profile, otherwise it SHALL be unselected for this Profile.

- Content: none
- Cardinality: 0-n
- Parent Elements: Profile
- Attributes: idref+, selected+
- Child Elements: none

The ’idref’ attribute MUST match the id attribute of a Group or Rule in the Benchmark, or the cluster id assigned to one or more Rules or Groups. The ’idref’ attribute values of sibling select element children of a Profile MUST be different.

6.2.7.51. <set-value>

This element specifies a value for a Value object. It MAY appear as part of a Profile; in that case it overrides the value property of a Value object. It MAY appear as part of a TestResult; in that case it supplies the value used in the test.

- Content: string
- Cardinality: 0-n
- Parent Elements: Profile, TestResult
- Attributes: idref+
- Child Elements: none

In the content of a Profile, the identifier given for the 'idref' attribute MAY be a cluster id, in which case it applies only to the Value item members of the cluster; in the context of a TestResult, the identifier MUST match the id of a Value object in the Benchmark. The 'idref' attribute values of sibling set-value element children of a Profile MUST be different.

6.2.7.52. <signature>

This element MAY hold an enveloped digital signature expressed according to the XML Digital Signature standard [6]. This element MUST contain exactly one element, Signature from the XML-Signature namespace.

- Content: elements
- Cardinality: 0-1
- Parent Elements: Benchmark, Rule, Group, Value, Profile, TestResult
- Attributes: none
- Child Elements: Signature* (in XML-Signature namespace)

At most one enveloped signature MAY appear in an XCCDF document. If multiple signatures are needed, others MUST be detached signatures.

The Signature element SHOULD contain exactly one Reference to the block enclosing the signature. The Reference URI SHOULD be a

relative URI to the Id attribute value of the enclosing object. For example, if the Id="abc", then the Reference URI would be "#abc".

6.2.7.53. <source>

The source element contains a URI indicating where a tailoring or benchmarking tool might obtain the value, or information about the value, for a Value object. XCCDF does not attach any meaning to the URI; it MAY be an arbitrary community or tool-specific value, or a pointer directly to a resource.

- Content: none
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: uri
- Child Elements: none

If several values for the source property appear, then they represent alternative means or locations for obtaining the value, in descending order of preference (i.e., most preferred first).

6.2.7.54. <status>

This element provides a revision or standardization status for a Benchmark, along with the date at which the Benchmark attained that status. It MUST appear at least once in a Benchmark object, and MAY appear once or more than once in any Item. If an Item does not have its own status element, its status SHALL be that of its parent element. The permitted string values for status are "accepted", "deprecated", "draft", "interim", and "incomplete".

- Content: string (enumerated choices)
- Cardinality: 0-n
- Parent Elements: Benchmark, Rule, Group, Value, Profile
- Attributes: date
- Child Elements: none

6.2.7.55. <sub>

This element represents a reference to a parameter value that MAY be set during tailoring. The element SHALL NOT have any content, and MUST have its single attribute, idref. The idref attribute MUST equal the id attribute of either a Value object or a plain-text definition.

- Content: none
- Cardinality: 0-n
- Parent Elements: description, fix, fixtext, front-matter, rationale, rear-matter, title, warning
- Attributes: idref+
- Child Elements: none

6.2.7.56. <target>

This element gives the name or description of a target system to which a Benchmark test was applied. It SHALL only appear as a child of a TestResult element.

- Content: string
- Cardinality: 1
- Parent Elements: TestResult
- Attributes: none
- Child Elements: none

6.2.7.57. <target-address>

This element gives the network address of a target system to which a Benchmark test was applied. It SHALL only appear as a child of a TestResult element.

- Content: string
- Cardinality: 0-n
- Parent Elements: TestResult

- Attributes: none
- Child Elements: none

6.2.7.58. <target-facts>

The TestResult object can hold an arbitrary set of facts about the target of a test. This element holds those facts, each one of which is a fact element. It is an OPTIONAL member of TestResult.

- Content: string
- Cardinality: 0-1
- Parent Elements: TestResult
- Attributes: none
- Child Elements: fact

6.2.7.59. <title>

This element provides the descriptive title for a Benchmark, Rule, Group, or Value. Multiple instances MAY appear with different languages (different values of the xml:lang attribute).

- Content: string
- Cardinality: 0-n
- Parent Elements: Benchmark, Value, Group, Rule, Profile, TestResult
- Attributes: xml:lang, override
- Child Elements: none

This element SHALL NOT contain XHTML markup.

The ’override’ attribute controls how the title property is inherited, if the Item in which it appears extends another Item.

6.2.7.60. <upper-bound>

This element MAY appear one or more times as a child of a Value element. It is used to constrain value input during tailoring, when the Value’s type is "number". It contains a number; values supplied

by the user for tailoring the Benchmark MUST be no greater than this number. This element MAY have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one upper-bound element appears as the child of a Value, at most one MAY omit the selector attribute.

- Content: number
- Cardinality: 0-n
- Parent Elements: Value
- Attributes: selector
- Child Elements: none

6.2.7.61. <value>

This string element is used to hold the value of a Value object. It MUST appear as the child of a Value element. This element MAY have a selector tag attribute, which identifies it for Value refinement by a Profile. This element MAY appear more than once, but at most one of the sibling instances of this element MAY omit the selector tag.

- Content: String
- Cardinality: 1-n
- Parent Elements: Value
- Attributes: Selector
- Child Elements: None

6.2.7.62. <version>

This element gives a version number for a Benchmark, Group, Rule, Value, or Profile. The version number content MAY be any string. This element MAY have a time attribute, which is a timestamp of when the Object was defined. This element MAY also have an update attribute, which SHOULD be the URI specifying where updates to the Object may be obtained.

- Content: string
- Cardinality: 1 (Benchmark), 0-1 (all others)

- Parent Elements: Benchmark, Group, Rule, Value, Profile
- Attributes: time, update
- Child Elements: none

6.2.7.63. <warning>

This element provides supplementary descriptive text for a Benchmark, Rule, Group, or Value. Multiple warning elements MAY appear; processing tools SHOULD concatenate them for generating reports or documents (see also Section 6.3).

- Content: mixed
- Cardinality: 0-n
- Parent Elements: Benchmark, Group, Rule, Value
- Attributes: xml:lang, override
- Child Elements: sub, xhtml elements*

This element is intended to convey important cautionary information for the Benchmark user (e.g., "Complying with this rule will cause the system to reject all IP packets"). Processing tools MAY present this information specially in generated documents.

6.3. Handling Text and String Content

This sub-section provides additional information about how XCCDF processing tools MUST handle textual content in Benchmarks.

6.3.1. XHTML Formatting and Locale

Some text-valued XCCDF elements MAY contain formatting specified with elements from the XHTML Core Recommendation.

Many of the string and textual elements of XCCDF are listed as appearing multiple times under the same parent element. These elements, listed below, MAY have an xml:lang attribute that specifies the natural language locale for which they are written (e.g., "en" for English, "fr" for French). A processing tool SHOULD employ these attributes when possible during tailoring, document generation, and producing compliance reports, to create localized output. An example of using the xml:lang attribute is shown below.

```
<cdf:Value id="web-server-port" type="number">
  <cdf:title>Web Server Port</cdf:title>
  <cdf:question xml:lang="en">
      What is the web server's TCP port?
  </cdf:question>
  <cdf:question xml:lang="fr">
      Quel est le port du TCP du web serveur?
  </cdf:question>
  <cdf:value>80</cdf:value>
</cdf:Value>
```

Multiple values for the same property in a single Item are handled differently, as described below. Multiple instances with different values of their xml:lang attribute SHALL be permitted; an Item with no value for the xml:lang attribute SHALL be taken to have the same language as the Benchmark itself (as given by the xml:lang attribute on the Benchmark element).

Elements	Inheritance Behavior
description, title, fixtext, rationale, question, front-matter, rear-matter	At most one instance per language; inherited values with the same language get replaced
warning, reference, notice	Multiple instances treated as an ordered list; inherited instances prepended to the list

The platform element MAY also appear multiple times, each with a different id, to express the notion that a Rule, Group, Benchmark, or Profile applies to several different products or systems.

6.3.2. String Substitution and Reference Processing

There are three kinds of string substitution and one kind of reference processing that XCCDF document generation and reporting tools MUST support.

1. XCCDF sub element - The sub element supports substitution of information from a Value object, or the string content of a plain-text definition. The formatting for a sub element reference to a Value object is implementation-dependent for document generation, as described in Section 5.3. Formatting for a sub element reference to a plain-text definition is very simple: the string content of the plain-text definition replaces the sub element.

2. XHTML object element - The object element supports substitutions of a variety of information from another Item or Profile, or the string content of a plain-text definition. To avoid possible conflicts with uses of an XHTML object that should not be processed specially, all XCCDF object references MUST be a relative URI beginning with "#xccdf:". The following URI values can be used to refer to things from an "object" element, using the "data" attribute:

o #xccdf:value:id - Insert the value of the plain-text block, Value, or fact with id id:. When a URI of this form is used, the value of the reference SHOULD be substituted for the entire "object" element and its content (if any). In keeping with the standard semantics of XHTML, if the id cannot be resolved, then the textual content of the "object" element SHOULD be retained.

o #xccdf:title:id - Insert the string content of the "title" child of the Item with id id. Use the current language value locale setting, if any. When a URI of this form is used, the title string SHOULD be substituted for the entire "object" element and its content (if any). In keeping with the standard semantics of XHTML, if the id cannot be resolved, then the textual content of the "object" element SHOULD be retained.

3. XHTML anchor (a) element - The anchor element MAY be used to create an intra-document link to another XCCDF Item or Profile. To avoid possible conflicts with uses of the XHTML anchor element that should not be processed specially, all XCCDF anchor references MUST be a relative URI beginning with "#xccdf:". The following URI values can be used to refer to things from an anchor ("a") element, using the ’href’ attribute:

o #xccdf:link:id - Create an intra-document link to the point in the document where the Item id is described. The content of the element SHOULD be the text of the link.

7. Security Considerations

In most situations XCCDF is not security sensitive. An XCCDF checklist may expose information about the configuration of a system that attackers can use. In these cases confidentiality would be desireable; however, there is no mechanism to encrypt information. If confidentiality is needed, use TLS or other forms of external encryption.

XML signature MAY be used within the XCCDF data model to add integrity and origin authentication. If you are using this for origin authentication, minimum key sizes SHOULD be RSA 1024 bit and SHA1. Implemetations SHOULD provide support for RSA 2048 bit or SHA256.

8. IANA Considerations

This document requires no action by IANA.

9. This draft defines common URLs and URNs for use within this data model as defined in Appendix A. Uniqueness and semantics for these URLs and URNs has been handled as part of XCCDF maintenance. Future standards based upon XCCDF would be required to establish IANA registries for these values. The URLs and URNs in Appendix A would be the initial values for those registries.Conclusions

The XCCDF specification defines a means for expressing security guidance documents in a way that should foster development of interoperable tools and content. It is designed to permit the same document to serve in several roles:

- Source code for generation of publication documents and hardcopy
- Script for eliciting local security policy settings and values from a user
- Structure for containing and organizing code that drives system analysis and configuration checking engines
- Source code for text to appear in security policy compliance reports
- A record of a compliance test, including the results of applying various rules

o Structure for expressing compliance scoring/weighting decisions.

XCCDF 1.1 was designed as a compatible extension of 1.0, based on suggestions from early adopters and potential users. Many features have been added, but every valid 1.0 document SHOULD be a valid 1.1 document, once the namespace is adjusted. The forward compatibility will help ensure that content written for XCCDF 1.0 will not be made obsolete by the 1.1 specification. XCCDF 1.1.4 is a minor update of the specification and schema of XCCDF 1.1, mainly to add clarity and fix problems identified by initial users.

Adoption of a common format should permit security professionals, security tool vendors, and system auditors to exchange information more quickly and precisely, and also permit greater automation of security testing and configuration checking.

10. References

10.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Marsh, J. and Orchard, D., "XML Inclusions (XInclude) Version 1.0", W3C Candidate Recommendation, April 2004.

[3] Baker, M. et al, "XHTML Basic", W3C Recommendation, December 2000.

[4] Biron, P. and Malhotra, A., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001.

[5] Buttner, A., and Ziring, N., "Common Platform Enumeration (CPE) - Specification, Version 2.2", MITRE Corporation, March 2009.

[6] Bartel, M. et al, "XML - Signature Syntax and Processing", W3C Recommendation, February 2002.

[7] Mell, P., Romanosky, S., and Scarfone, K., "A Complete Guide to the Common Vulnerability Scoring System Version 2.0", FIRST, June 2007.

[8] Davis, M., "Unicode Regular Expressions", Unicode Technical Recommendation No. 18, version 9, January 2004.

[9] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

10.2. Informative References

[10] Quinn, S. et al, "National Checklist Program for IT Products - Guidelines for Checklist Users and Developers", NIST Special Publication 800-70 Revision 1, September 2009.

[11] "OVAL - The Open Vulnerability and Assessment Language", The MITRE Corporation, September 2006.

[12] Johnston, P. and Powell, A., "Guidelines for Implementing Dublin Core in XML", DCMI, April 2003.

[13] Hillmann, D., "Using Dublin Core", DCMI, August 2003.

11. Acknowledgments

This draft is based on the NIST Interagency Report (IR) 7275 revision 3 authored by Neal Ziring of the NSA and Stephen D. Quinn of NIST. This document is the result of the collective effort of a large number of people including the following individuals who contributed to the initial definition and development of XCCDF: David Proulx, Mike Michnikov, Andrew Buttner, Todd Wittbold, Adam Compton, George Jones, Chris Calabrese, John Banghart, Murugiah Souppaya, John Wack, Trent Pitsenbarger, and Robert Stafford.

Peter Mell and Matthew Wojcik contributed to Revisions 1, 2, and 3 of the NISTIR 7275. Karen Scarfone provided editorial comments and changes to all historic revisions and to this draft. Ryan Wilson of Georgia Institute of Technology also made substantial contributions. Thanks also go to the DISA Field Security Office (FSO) Vulnerability Management System (VMS)/Gold Disk team for extensive review and many suggestions.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Pre-Defined URIs

The following URLs and URNs are defined for XCCDF 1.1.

A.1. Long-Term Identification Systems

These URIs may appear as the value of the system attribute of an ident element (see page 59).

http://cve.mitre.org/ - MITRE's Common Vulnerabilities and Exposures - the identifier value should be a CVE number or CVE candidate number.

http://cce.mitre.org/ - This specifies the Common Configuration Enumeration identifier scheme.

http://www.cert.org/ - CERT Coordination Center - the identifier value should be a CERT advisory identifier (e.g. "CA-2004-02").

http://www.us-cert.gov/cas/techalerts/ - US-CERT technical cyber security alerts - the identifier value should be a technical cyber security alert ID (e.g., "TA05-189A")

http://www.kb.cert.org/ - US-CERT vulnerability notes database - the identifier values should be a vulnerability note number (e.g., "709220").

http://iase.disa.mil/IAalerts/ - DISA Information Assurance Vulnerability Alerts (IAVA) - the identifier value should be a DOD IAVA identifier.

A.2. Check Systems

These URIs may appear as the value of the system attribute of a check element (see p. 50).

http://oval.mitre.org/XMLSchema/oval - MITRE's Open Vulnerability Assessment Language (see [11]).

http://www.cisecurity.org/xccdf/interactive/1.0 - Center for Internet Security interactive query check system, used for asking the user questions about the target system during application of a security guidance document.

Other check systems are being developed and used to support many use cases. This listing includes publicly released check systems available at the release of this draft.

A.3. Scoring Models

These URIs may appear as the value of the system attribute on the model element or a score element (see pp. 62 and 71).

urn:xccdf:scoring:default - This specifies the default (XCCDF 1.0) scoring model.

urn:xccdf:scoring:flat - This specifies the flat, weighted scoring model.

urn:xccdf:scoring:flat-unweighted - This specifies the flat scoring model with weights ignored (all weights set to 1).

urn:xccdf:scoring:absolute - This specifies the absolute (1 or 0) scoring model.

A.4. Target Platform Facts

The following URNs should be used to record facts about an IT asset (target) to which a Benchmark TestResult applies (see pages 49 and 75).

urn:scap:fact:asset:identifier:mac - Ethernet media access control address (should be sent as a pair with the IP or IPv6 address to ensure uniqueness)

urn:scap:fact:asset:identifier:ipv4 - Internet Protocol version 4 address

urn:scap:fact:asset:identifier:ipv6 - Internet Protocol version 6 address

urn:scap:fact:asset:identifier:host_name - Host name of the asset, if assigned

urn:scap:fact:asset:identifier:fqdn - Fully qualified domain name

urn:scap:fact:asset:identifier:ein - Equipment identification number or other inventory tag number

urn:scap:fact:asset:identifier:pki: - X.509 PKI certificate for the asset (encoded in Base-64)

urn:scap:fact:asset:identifier:pki:thumbprint - SHA.1 hash of the PKI certification for the asset (encoded in Base-64)

urn:scap:fact:asset:identifier:guid - Globally unique identifier for the asset

urn:scap:fact:asset:identifier:ldap - LDAP directory string (distinguished name) of the asset, if assigned

urn:scap:fact:asset:identifier:active_directory - Active Directory realm to which the asset belongs, if assigned

urn:scap:fact:asset:identifier:nis_domain - NIS domain of the asset, if assigned

urn:scap:fact:asset:environmental_information:owning_organization - Organization that tracks the asset on its inventory

urn:scap:fact:asset:environmental_information:current_region - Geographic region where the asset is located

urn:scap:fact:asset:environmental_information:administration_unit - Name of the organization that does system administration for the asset

urn:scap:fact:asset:environmental_information:administration_poc:title - Title (e.g., Mr, Ms, Col) of the system administrator for an asset]

urn:scap:fact:asset:environmental_information:administration_poc:e-mail - E-mail address of the system administrator for the asset

urn:scap:fact:asset:environmental_information:administration_poc:first_name - First name of the system administrator for the asset

urn:scap:fact:asset:environmental_information:administration_poc:last_name - Last name of the system administrator for the asset

A.5. Remediation Systems

The URIs represent remediation sources, mechanisms, schemes, or providers. They may appear as the system attribute on a fix element (see p. 56).

urn:xccdf:fix:commands - This specifies that the content of the fix element is a list of target system commands; executed in order, the commands should bring the target system into compliance with the Rule.

urn:xccdf:fix:urls - This specifies that the content of the fix element is a list of one or more URLs. The resources identified by the URLs should be applied to bring the system into compliance with the Rule.

urn:xccdf:fix:script:{LANGUAGE} - A URN of this form specifies that the content of the fix element is a script written in the given language as specified by the {LANGUAGE} term. Executing the script should bring the target system into compliance with the Rule. The following {LANGUAGE} terms are pre-defined:

- sh - Bourne shell
- csh - C Shell
- perl - Perl
- batch - Windows batch script
- python - Python and all Python-based scripting languages
- vbscript - Visual Basic Script (VBS)
- javascript - Javascript (ECMAScript, JScript)
- tcl - Tcl and all Tcl-based scripting languages

urn:xccdf:fix:patch:vendor - A URN of this form specifies that the content of the fix element is a patch identifier, in proprietary format as defined by the vendor. The vendor string should be the DNS domain name of the vendor, as defined in the CPE 2.2 specification [5]. For example, for Microsoft Corporation, the DNS domain is "microsoft.com", and the CPE vendor name would be "microsoft".

Authors' Address

David Waltermire
NIST
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930

Email: david.waltermire@nist.gov