An Introduction to the Common Configuration Enumeration

Version: 1.7
Date: July 24, 2008
Author: David Mann

Table of Contents

1 Introduction

The Common Configuration Enumeration (CCE™) assigns unique identifiers to configuration statements in order to facilitate fast and accurate correlation of configuration statements present in disparate domains. In this way, it is similar to other data standards such as MITRE's Common Vulnerability and Exposure (CVE®) List[1], which assigns identifiers to publicly known system vulnerabilities.

The purpose of this document is to provide an overview of CCE including its basic concepts, its oversight, and its creation process. The introduction contains a summary of the issues discussed in this document. The second section describes how the CCE project is managed and governed. The third section discusses what a CCE entry is (also called "CCEs") and the basic concepts involved in their definitions. The fourth section describes the five enterprise use cases that CCE is envisioned to support. The fifth section gives more detail on the CCE data model. The sixth section explores several assumptions about the CCE creation process and their implications as they relate to inter-organizational cooperation to create and publish CCE data. And the seventh and final section identifies three primary use cases that need to be supported by any CCE creation process along with some suggestions on how these use cases might be supported.

In this document, we will refer to an operating system (OS) or application as a "platform." Platforms typically provide a set of configuration options which will have various effects on the performance and behavior of the platform. CCE Identifiers (CCE-IDs) provide common identifiers for these configuration controls.

The purpose of this document is to describe the concepts and processes involved in the creation and basic editing of CCE entries. The CCE List is available for public use on the CCE Web site at http://cce.mitre.org.

2 Moderation

CCE is currently managed by the MITRE Corporation[2]. This section gives a brief overview of how the CCE effort is moderated and managed.

2.1 MITRE and Information Assurance Data Standards

MITRE is a not-for-profit corporation, chartered to work solely in the public interest. MITRE operates three U.S. Federally Funded Research and Development Centers (FFRDCs). The FFRDC sponsored by the Department of Defense (DoD) specializes in command, control, communications, and intelligence systems. The FFRDC sponsored by the Federal Aviation Administration focuses on the development of a safe, efficient, worldwide air traffic management system. MITRE's newest FFRDC, sponsored by the Internal Revenue Service (IRS), provides technical and program management advice to the IRS and other Treasury Department agencies in support of enterprise systems modernization.

[1] http://cve.mitre.org/cve
[2] www.mitre.org

An FFRDC is a unique organization that assists the U.S. government with scientific research and analysis, development and acquisition, and/or systems engineering and integration. FFRDCs address long-term problems of considerable complexity, analyze technical questions with a high degree of objectivity, and provide creative and cost-effective solutions to government problems. Working in the public interest, FFRDCs operate as long-term strategic partners with their sponsoring government agencies. In order to ensure the highest levels of objectivity, FFRDCs are organized as independent entities with limitations and restrictions on their activities. This unique standing permits FFRDCs to approach difficult problems while maintaining a long-term perspective. Since FFRDCs are prohibited from manufacturing products, competing with industry, or working for commercial companies, industry and government confidently provide them with sensitive information.

It is this FFRDC role that has led MITRE to collaborate with industry and government to create several data standards efforts in the information security or information assurance (IA) industry. Launched in 1999, the CVE List is the oldest of these standards. Other MITRE-managed IA data standards include the Open Vulnerability and Assessment Language (OVAL®)[3], the Common Weakness Enumeration (CWE™)[4], the Common Platform Enumeration (CPE™)[5], and the Common Attack Pattern Enumeration and Classification (CAPEC™)[6]. A complete list of MITRE-managed IA data standards along with a list of related government and industry standards can be found on the Making Security Measurable Web site at http://measurablesecurity.mitre.org.

2.2 Current Management of CCE

In fulfillment of its FFRDC charter, MITRE seeks to employ a standards development and moderation approach for its data standards that will (1) reflect the long-term strategic needs of our governmental sponsors, and (2) be useful and adopted by industry while remaining non-competitive with industry. It should be emphasized that these goals are not in conflict with each other. It is in the public's and government's best interest for CCE (and the other data standards) to develop and mature in a way that provides real value for industry and that fosters widespread adoption within the industry. To meet this goal, MITRE employs a range of different standards development and moderation practices which are tailored to reflect the different needs of the different sub-industries and the different maturity levels of the standards effort.

At the time of writing, MITRE moderates the CCE Working Group. The Working Group's membership is made up of interested stakeholders from government, commercial tool vendors, end users of tools, and academics. The Working Group discusses CCE on an email list, on monthly teleconferences, and occasionally face-to-face meetings. MITRE facilitates discussion of topics on the mail list and moderates the teleconferences, and face-to-face meetings. MITRE seeks and considers the input of all participants in the Working Group and strives to help the group arrive at consensus on decisions. A part of the consensus building process includes informal votes, or straw ballots. When clear consensus is not apparent, MITRE makes decisions on the standard with close consultation with our government sponsors with the goal of ensuring

[3] http://oval.mitre.org
[4] http://cwe.mitre.org
[5] http://cpe.mitre.org
[6] http://capec.mitre.org

the maximum benefit to the public interest. At the time of this writing, CCE is sponsored by the Vulnerability Analysis & Operations Group of the U.S. National Security Agency (NSA)[7].

2.3 Future Management of CCE

As CCE matures the management of CCE may also evolve. There are three kinds of change that may occur over time. First, MITRE in close coordination with the CCE Working Group may alter its direct moderation style to reflect changing needs within the industry. Possible changes might include the creation of more formalized oversight or advisory groups (e.g., an Editorial Board), more formalized membership and voting procedures, and more formalized comment and change control processes for CCE documents and content.

The second type of possible change for CCE applies equally to all MITRE standards. Over the course of many years, it is not uncommon for a particular standard to receive sponsorship from different specific government sponsors. MITRE works with its strategic government partners to ensure that such changes maintain the integrity of the standards efforts.

Third, in its FFRDC role, MITRE has a formalized technology transfer process whereby technologies and capabilities can be transferred to other organizations when it is in the public interest to do so. Transference of CCE to a government or industry standards body is a possible long-term outcome.

3 CCE BASICS

This section will provide a basic understanding of CCE entries. We address three primary questions:

- What is a CCE entry?
- When does CCE consider configuration statements to be the same?
- When does CCE consider configuration statements to be different?

3.1 Basic Concepts

Configuration statements can be found in a variety of repositories such as security guides, benchmarks, vendor guidance and documentation, configuration assessment and management tools, and consolidated reporting systems.

CCE facilitates fast, accurate correlation between elements in these domains by assigning a unique name to each atomic configuration statement.

Example:

- CCE-3177-3
- Definition: The “account lockout threshold” setting should be configured correctly.
- Parameters: Number of attempts.

[7] http://www.nsa.gov

In general, the scope of CCE-IDs is determined by the ability of an automated process to check the parameter of a particular configuration statement. If a configuration statement can be verified by an assessment tool or applied by configuration management system, it should be assigned a CCE-ID.

Should Get CCEs:

- The required permissions for the directory %SystemRoot%\System32\Setup should be assigned.
- The "account lockout threshold" setting should be configured correctly.
- The startup type of the Remote Shell service should be set correctly.

The examples above are all able to be verified automatically and can be set and checked using automated scripts, programs, or OS functions. These do not require any human effort in order to verify or interpret and can all be assigned CCE-IDs.

Should NOT Get CCEs:

- Never add user passwords to the users.conf file through a text editor.
- The computer is stored in a locked room.
- Use strong passwords.

There are several reasons we might exclude statements from receiving a CCE-ID. First, we exclude statements that are effectively impossible to verify (without finer accountability controls within the operating systems), for example verifying the methods that a user might use to edit users.conf would require a substantial effort and is non-trivial without finer accountability systems within OSs.

Second, we exclude statements that require substantial human interpretation in order to be technically implemented. For example, the statement "Use strong passwords" is ambiguous and different analysts will interpret its technical meaning differently. For Microsoft Windows XP, CCE would replace the statement "Use strong passwords" with the five statements as shown below.

Example: Use strong passwords.

- The "minimum password age" setting should meet minimum requirements. (CCE-2439-8)
- The "minimum password length" setting should meet minimum requirements. (CCE-2981-9)
- The "password must meet complexity requirements" setting should be configured correctly. (CCE-2735-9)
- The "enforce password history" setting should meet minimum requirements. (CCE-2994-2)
- The "store password using reversible encryption for all users in the domain" setting should be configured correctly. (CCE-2889-4)

Led by MITRE, CCE is developed through the collaboration of industry experts via the CCE Working Group that is representative of a variety of configuration guidance domains. CCE,

which assists configuration guidance authors and technical implementers by providing lists of already refined statements to aid the clarification of configuration statements, represents the collective work of configuration authors across the industry. By documenting and using the "wisdom of the crowds" everyone benefits from configuration statements that are less ambiguous and more readily meaningful from one group to another.

To document this industry consensus, the CCE Working Group produces:

- CCE Content Creation Documentation - A set of documents that describe the CCE creation and submission process that can be used by configuration guidance authors and technical implementers to allow content to be written in manner that facilitates better technical implementation.
- CCE Lists - Sets of common configuration statements with unique identifiers (names), definitions, parameters, and technical mechanism enumerations.

3.2 Comparable Statements

CCE treats some configuration statements as being essentially the same, despite apparent differences. Configuration statements that are considered to be the same from the viewpoint of CCE are referred to as comparable statements. There are two primary ways in which statements are comparable and these are captured by CCE's parameters and technical mechanisms.

3.2.1 Parameters

With respect to parameters, consider the following configuration statements.

Example Configuration Statements:

- Source: DISA Gold Disk Tool for Windows 2000
  - Statement: Account logon lockout threshold = 3

- Source: CIS Level 1 Benchmark for Windows 2000
  - Statement: Account logon lockout threshold = 50

CCE considers these two statements to be comparable as they are both about the same setting, the "Account logon lockout threshold". They differ only in the particular value for that setting asserted by the different guide authors. To capture this, CCE treats the number of maximum logon attempts as a parameter. The resulting CCE entry follows:

Example CCE:

- CCE-3229-2
- Definition: The "account lockout threshold" policy should meet minimum requirements.
- Parameter: Number of attempts.

It is important to note that the definition is written to be agnostic towards a specific recommended parameter value. Instead, CCE attempts to capture all potential values a configuration statement might encompass in a set of parameters.

CCE parameters identify the possible values a configuration control might have at a conceptual level.

3.2.2 Technical Mechanisms

The second manner in which configuration statements are considered to be comparable within CCE is by technical mechanisms. CCE technical mechanisms describe the underlying technical control used to affect the desired change on the system as indicated by the configuration control. Consider the following examples taken from Windows Vista.

Example Technical Mechanisms:

- Windows Registry Key:
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\Logging\LogDroppedPackets
- GPO Template:
  Computer Configuration\Administrative Templates\Network\Network Connections\Windows Firewall\Standard Profile\Windows Firewall: Allow Logging - Log Dropped Packets
- Windows Firewall Settings:
  Control Panel\Windows Firewall\Advanced\Security Logging\Logging Options\ Log dropped packets

These examples illustrate that there are (at least) three methods for setting the "Log Dropped Packets" option in the Windows Firewall: by making direct registry key edits, by using the GPO editor, or by using the Windows Firewall tool interface. While there are subtle reasons why an administrator would choose one method over another, there is agreement that each of these methods is accomplishing a comparable goal. Within CCE, these statements represent three comparable technical mechanisms for the same CCE.

Example CCE:

- CCE-3260-7
- Definition: The "Log Dropped Packets" option for the Windows Firewall should be configured correctly for the Domain Profile.
- Parameter: Enabled or Disabled.
- Technical Mechanisms:
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\Domain Profile\Logging\LogDroppedPackets
  - Computer Configuration\Administrative Templates\Network\Network Connections\Windows Firewall\Domain Profile\Windows Firewall: Allow Logging - Log Dropped Packets
  - Control Panel\Windows Firewall\Advanced\Security Logging\Logging Options\ Log dropped packets

3.3 Non-Comparable Statements: Platform Groups

While CCE treats some configurations as being comparable despite their apparent differences, CCE also distinguishes between other configuration statements as being different, despite their apparent similarity. We refer to such statements as being non-comparable. There are two primary ways in which similar statements are considered to be non-comparable within CCE: differing conceptual security models and differing platform group associations.

3.3.1 Differing Security Models

Consider the example of a configuration statement regarding the file permissions on a particular system file for a program that can be installed on both UNIX and Windows systems. In this case, one might try to make a single configuration statement despite the fact the file would be stored in different native file systems such as NTFS or NFS. Similarly, consider the case when the file of interest is stored in a directory that is shared on the network via SMB. In this case, it would be insufficient to make a single statement that applies to both the native file system and to the SMB share.

CCE observes that each of the file systems mentioned, NFS, SMB and NTFS, all have different security models and manage rights in fundamentally different ways. For this reason, CCE assigns different CCE-IDs for each configuration statement; one for each applicable file system.

A more detailed discussion about separating CCEs based on differing security models can be found in the "CCE Editorial Policies" document posted in the CCE List section of the CCE Web site at http://cce.mitre.org/lists/cce_list_editorialpolicies.html.

3.3.2 Platform Groups

It is not uncommon for features and configuration controls to be shared across different platforms that share portions of their code base. This often happens within platform families from a single vendor or across different vendors with platforms derived from similar legacy code bases (e.g., the UNIX and Linux variants).

CCE distinguishes between otherwise similar (or even identical) configuration statements when they are made for different major platforms. Consider the following example.

Example:

- The required auditing for %SystemDrive% directory should be enabled.

This statement applies equally to both Windows 2000 and Windows XP. But CCE recognizes both Windows 2000 and Windows XP as different "platform groups" and thus assigns different CCE-IDs to this statement, one for each platform group.

The CCE platform group roughly identifies the operating system or application to which the CCE entry applies. CCE platform groups are a set of high-level "buckets" that imply a particular CCE is "related to" the OS or application named by the platform group. These groups are not

definitive declarations of a CCE entry's relation to a particular platform. They are meant for human interpretation.

Some examples of CCE Platform Groups include:

- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows Server 2003
- Microsoft Internet Explorer 7
- Microsoft Office 2007
- Red Hat Enterprise Linux 4
- Red Hat Enterprise Linux 5
- Sun Solaris 9
- Sun Solaris 10

The intent of CCE platform groups is to align CCE with how platform assertions are made within the contexts of two of the central use cases for CCE. The first is the issuing of security configuration guidance. We observe that in common practice, security guides are issued for different platforms at the same level of granularity that we have defined CCE platform groups. The second primary use case is sets of checks or functionality in configuration audit and configuration management tools. Again, we observe that the convention in common practice among configuration audit and management tools is to present sets of checks and other features in groups of functionality that correspond to the level of granularity at which we have defined CCE platform groups.

As we have observed that the common practice of members of the various communities of practice separate their configuration statements by the applicable platform, CCE has decided to make platform groups a non-comparable entity. Each CCE entry will be assigned a single platform group. Similar configuration statements that cross the CCE platform groups will be assigned separate CCE-IDs. We recognize that different use cases will want to track the relationships between CCEs and platforms in different ways. Thus we emphasize that the definition of a CCE platform group and the association of CCEs with platform groups is the sole discretion of the CCE project. We must emphasize that the exact meaning of "Microsoft Windows XP" may differ from context to context. The definition of CCE platform groups by the CCE project localizes the meaning of that statement within the CCE definition context. This meaning may not apply to all contexts.

In particular, we emphasize that the inclusion of a CCE in a platform group does not imply that the CCE applies too all variants of that platform. It is impossible for CCE, the CCE Working Group, and guide authors to verify all configuration controls against all possible variants of a platform at the time of CCE creation. In fact, it is absolutely impossible to know about the exact behavior of future versions of the platform before they are released. For example, the Windows XP platform group contains CCEs for configuration controls that only become available after the installation of Service Pack 2.

Putting this another way, CCE platform groups document the degree to which CCE distinguishes between platforms. Windows 2000 is distinguished from Windows XP, but Windows XP pre-Service Pack 2 and Windows XP post-Service Pack 2 are not distinguished.

In general, assignment of CCE entries to CCE platform groups will be as follows:

- If the statement comes from an OS-oriented source and concerns an OS construct, it will be assigned the associated OS platform group.
- If the statement comes from an application-oriented source and concerns the application itself, it will be assigned the associated application platform group.
- If the statement comes from an OS-oriented source and clearly concerns an application for which no CCE platform group has been created (e.g., Microsoft Telnet, Bind), then it will be assigned to the OS platform group.
- If the statement comes from an OS-oriented source and clearly concerns an application for which a CCE platform group has been created (e.g., Microsoft Internet Explorer), then it will be assigned to the corresponding application platform group.
- If the statement comes from an application-oriented guide and clearly concerns a configuration control within the base OS, then it will be assigned the to the OS application group.
- All other assignments will be at the discretion of the CCE Moderator (currently MITRE).

4 CCE Use Cases

The creation of CCE has been motivated by five separate use cases within the context of security configuration management in the enterprise. The core use case is the internal configuration management lifecycle that is conducted within an enterprise. We describe this use case as a four part process: system design, pre-deployment testing, deployment and assessment, and remediation. The remaining four CCE use cases can be seen as spokes emanating from the core cycle as they all relate to coordination and communication between some part of the configuration management lifecycle and some set of stakeholders. These four use cases include: system design, configuration audit, consolidated audit reporting, and regulatory compliance. Each of these use cases is described in further detail in this next section.

A common theme among all of these use cases is the use of CCEs to facilitate faster and more accurate correlation of configuration information across different, but closely related practices. Each of the practices have their own processes, procedures, and domain expertise. Additionally, members of each of these practices share information across their domain boundaries with other practices as part of their work. As work is conducted across domains, sometimes a set of commonly shared “objects” will emerge that allow the different practices to cooperate with each other and to coordinate their actions. These objects are sometimes called “boundary objects.” We believe that CCE-IDs can act as boundary objects in each of the five identified use cases.

The concept of boundary objects was introduced by Geoffrey Bowker and Susan Star in *Sorting Things Out*[8]:

> Boundary objects are those objects that both inhabit several communities of practice and satisfy the informational requirements of each of them. Boundary objects are thus both plastic enough to adapt to local needs and constraints of the several parties employing them, yet robust enough to maintain common identity across sites. They are weakly structured in common use and become strongly structured in individual-site use. These objects may be abstract or concrete. Star and Griesemer (1989) first noticed the phenomenon in studying a museum, where specimens of dead birds had very different meaning to amateur bird watchers and professional biologists, but "the same" bird was used by each group. Such objects have different meaning in different social worlds but structure is common enough to more than one world to make them recognizable, a means of translation. The creation and management of boundary objects is a key process in developing and maintaining coherence across intersecting communities.

An excellent example of boundary objects that most users should be familiar with are the International Standard Book Numbers (ISBN) used to identify books. In the book world there are many different communities of practice, such as book publishers, retailers, and libraries, each have their own processes and procedures to track and manage information about books. We might expect two book publishers to have similar practices to have the same billing and inventory systems. But we would not expect that an inventory system tailored for book publication would be usable in any way by a library. The semantic or schematic meaning of "book" is different for these communities of practice. But despite these different localized definitions of books, the practices can coordinate their actions using ISBNs. CCEs are designed to foster this kind of cross-boundary coordination of action for the different practices within the context of security configuration management.

In the following section, we will discuss five primary use cases for CCE entries. In each case, the CCE entries act as boundary objects between two different but related communities and provides them with a mutually recognized identification of a configuration issue.

4.1 The Configuration Management Lifecycle

This use case involves the communication and coordination of work among four (or more) groups involved in configuration management within an organization including system design, pre-deployment system testing, system provisioning and deployment, and configuration audit and remediation.

[8] Bowker, G. C., and Star, S. L. *Sorting Things Out: Classification and Its Consequences (Inside Technology)*, MIT Press, Boston, 1999.

4.1.1 Scenario

Company A plans on deploying a set of new systems on their network. To accomplish this, four (or more) groups must coordinate their action in the context of a broader configuration management lifecycle. First, the system designers must design the system to be compliant with the set of requirements that may include internal security policies, externally defined security requirements, and operational requirements. Second, system testers will build one or more systems according to the design to test the system's operational readiness. Third, the system deployment group will be responsible for deploying the tested system on the network. Finally, the assessment and management group (which may be further separated) is responsible for testing to ensure that systems remain compliant with the agreed upon configuration settings and for making changes as needed.

Typically, configuration standards for systems are captured in internally authored documents or spreadsheets. While there is a movement towards more automated expression of configuration standards these automated solutions are typically focused primarily on a single part of the configuration management lifecycle and are thus inaccessible to groups in other parts of the lifecycle. Examples of these emerging technologies include policy management products, system provisioning systems, network management capabilities (e.g., active directory group policy objects), and configuration benchmarking and audit tools.

4.1.2 Problem

In order for the different groups in the configuration management lifecycle to coordinate their actions, they need to be able to efficiently and accurately identify specific configuration controls with each other. Without CCE-IDs, these groups must rely on the "names" presented to them in the context of different written documents (e.g., security configuration guides) and different security and IT tools (e.g., Group Policy Objects (GPOs)). While these names are often similar they are rarely the same and two kinds of errors are typically encountered. First, configuration issues that are the same but named slightly differently in different contexts are not recognized as being the same. Second, issues that are closely related but different are not recognized as being different due to the similarities of their names.

As a result, the internally authored written configuration standard is not effective in coordinating the actions of the different groups within the configuration management lifecycle. Mapping errors are made at each step of the process and the groups must spend a large amount of analysis time to resolve the discrepancies. The different groups need a mechanism that will allow them to quickly and accurately collaborate across each step of the lifecycle.

4.1.3 How CCE Helps

By annotating their internally defined configuration standards with CCE-IDs, the groups involved in different parts of the configuration management lifecycle are better able to communicate with each other and coordinate their actions. At each step of the process, a given group can use the CCE-IDs to find more information on the configuration issue by accessing configuration information sources with which they are familiar. In this way, they can more quickly and accurately map the configuration standard into their own localized context and tool set.

In this scenario, CCE-IDs act as boundary objects across the boundaries of the different practices within the configuration management lifecycle. While each individual practice (design, testing, deployment, audit, and remediation) all deal with the concepts of configuration controls, the precise naming or articulation of those controls typically differs from practice to practice. The CCE-IDs are shareable objects that can be easily mapped into each practice.

4.2 Guide Document Authoring and System Design

This use case involves the cross-boundary coordination and communication that takes place between configuration guide authors and system designers.

4.2.1 Scenario

A configuration guide author has been tasked with writing a guidance document for a particular platform. The deliverable produced will be a human-readable prose document that is provisioned either as a static document (e.g., PDF, MS-Word) or as a dynamically allocated document (e.g., HTML, XML, XCCDF[9]). The author may work for the primary vendor organization that sells the platform (e.g., Apple, Microsoft, Red Hat, Sun) or an organization that is responsible for providing guidance for large populations (e.g., Center for Internet Security (CIS)[10], U.S. Defense Information Systems Agency(DISA)[11], National Institute of Standards and Technology (NIST)[12], NSA.

At the same time, a system designer has been tasked to design a new (or update an existing) configuration standard (template) for a system to be deployed within their organization. The designer is required to consider the security implications of each configuration control to ensure that the system complies with the company's security policies. Additionally, she is required to document if and how the design complies with industry best practices. To achieve this, she will read security configuration guides published by the platform vendor and by relevant third parties. She will also review information on particular configuration settings found on Web sites maintained by the vendor and in discussion forums. The end product that the system designer will produce will be a document that describes the configuration settings to be applied. She will, in most cases, also be required to document the references from the relevant best practices documents in order to justify her decisions.

4.2.2 Problems

One of the primary features of security configuration guides is their reliance on prose as the primary means of communication. This makes these documents human readable. It also allows the author to provide generalized guidance that should be human interpreted in order to be applied to a particular technical implementation or within a particular operational setting. For example, the generalized guidance to "use strong passwords" is valid guidance but it must be interpreted by a human in order to be applied within a particular context since there are many aspects of password strength. On the other hand, the very strength of human prose that allows it to be useful for describing ambiguous concepts also allows for different readers to interpret the

[9] Extensible Configuration Checklist Description Format (XCCDF); http://nvd.nist.gov/xccdf.cfm
[10] http://www.cisecurity.org
[11] http://www.disa.mil
[12] http://nvd.nist.gov

guidance in different ways. Not all technical readers will identify the same list of password attributes when interpreting “use strong passwords”.

Increasingly, security guide authors are beginning to augment their human prose with specific technical line items that enumerate the lists of corresponding technical controls. While this helps to disambiguate portions of the human prose, this is not a complete solution for two reasons. First, lacking CCE-IDs, the configuration guides that the system designer consults all refer to individual configuration settings with their own terminology, which means that it is common for the same issue to be identified with different names in different documents. Second and closely related, due to frequency of related sets of functionality, it is not uncommon for related but significantly different configuration controls to be “named” in nearly the same manner. That is, a common error occurs when the system designer attempts to reconcile guidance from two different sources about what she believes to be about the same configuration control (due to their similar names) when, in fact, they are about different controls. Third, owing to the fact that many configuration controls are themselves built upon other sub-systems and system calls, it is not uncommon for different security guides to annotate their prose by attempting to enumerate system controls at different levels of abstraction or granularity, which further confuses the system designer when she attempts to reconcile the two guidance documents.

The lack of a common way to identify the configuration controls means that the system designer must search the documents either by hand or using error-prone keyword searches if she has an electronic copy of the document available. As a result, she must spend a significant portion of her time attempting to locate the applicable section within the guidance document. The similarity in the naming practices of similar but different controls means that sometimes she must spend additional analysis time to determine if sections from two different documents are, in fact, referring to the same controls. The lack of common identifiers for configuration controls also means that she must rely on error-prone keyword searches when consulting online resources maintained by the vendor or when searching the Web for addition information. The result is longer analysis times during her research and more errors when attempting to integrate information from multiple sources.

4.2.3 How CCE Helps

Written security guides that are annotated with CCE-IDs allow the system designer to quickly and accurately identify which sections of the document relate to the configuration control with which she is concerned. The use of CCE-IDs to augment the human prose makes is more likely that the enumerations of technical controls that appear in two different documents are more likely to be expressed at the same level of abstraction (the level of abstraction of the corresponding CCE-IDs), which makes it more likely that the guidance documents are conceptually comparable. Additionally, the use of CCE-IDs in multiple security guides means that system designer can quickly and accurately correlate the information from both guides despite the fact that the guides are formatted differently and may refer to the same issues with slightly different proprietary names. The use of CCE-IDs allows her to quickly identify and ignore those sections that do not relate to the control she is interested in, despite its similar “name.” This reduces her analysis time and makes it more likely that she will avoid mistakes made from attempting to incorporate erroneous information.

The use of CCE-IDs in online support facilities provided by the vendor and other on-line resources including discussion forums allow the system designer to quickly and accurately find additional relevant information on the Internet. This type of increased findability can be witnessed today with the use of CVE Identifiers.

Regarding the use of CCE-IDs as boundary objects in this scenario, we note that system design and security guide authoring are different but related practices. While is common for security guide authors to have had experience in system design and administration, the process of creating guides is different from the process of designing systems. To be effective, both security guide authors and system designers need to collaborate, albeit indirectly through the publication and reading of the written security guides. While a particular configuration control will have a different form of expression in guides compared to system definition, they are closely related. The CCE-ID is plastic enough to be usable in both domains, allowing the work of configuration guidance and system design to be effectively related.

It should be noted that the assignment of CCEs on a per-platform group basis facilitates the inclusion of CCEs in the security guides and online help systems. Typically, these resources are compiled and created by groups of experts on a per platform group basis. By issuing CCEs on per platform basis, we make CCEs more in line with the existing practices within the security guidance domain.

4.3 Configuration Tool Configuration

The following use case concerns the communication and coordination between creators of configuration management tools and end users of those tools. This use case can be seen as a single component of the configuration management lifecycle use case described above. However, this use case emphasizes the external cross boundary coordination between the vendor and the customer organization, whereas the configuration management lifecycle emphasizes the internal cross boundary coordination between groups within the organization.

4.3.1 Scenario

A vendor produces a tool targeted to a portion of the configuration management lifecycle. Some of the most commonly used products consist of configuration management tools that enforce or change configuration settings on end systems (e.g., MS Active Directory Group Policy Objects) and configuration audit tools that are used to test the configuration status of end systems. As the vendor creates the product, they strive to represent configuration issues in their product in a manner that will be the most useful and comprehensible to their customers, where their “customer” includes not only the end users of their product but also stakeholders within the customer’s organization that are involved with other aspects of the configuration management lifecycle and must coordinate and communicate with both the product and end users of the product. The vendor is faced with the challenge of how to present names or identifiers for configuration controls or configuration checks that will allow end users to understand the meaning of a single control or to find a particular control in the set of controls.

An IT administrator who is responsible for the deployment and operational use of a configuration management tool has been tasked with the responsibility for tuning the tool to implement a step

in the configuration management lifecycle. As a part of the lifecycle, the administrator takes as input some set of configuration controls. This may be a set of controls to be deployed or enforced using a configuration management tool or a set of controls to be audited for using a configuration assessment tool. To successfully complete this task the administrator must non-ambiguously map the input set of controls to specific checks or functional capabilities within the tool. In most cases, the administrator will also be required to document the mapping between the written configuration document and the specific checks within the tool.

4.3.2 Problem

Lacking CCE-IDs, the tool vendor must name individual checks or controls within their product that are based on their own proprietary naming conventions. These proprietary names are typically related to but still different from other names for the same configuration controls used in other tools or documents. In fact, they may choose to intentionally differentiate their own proprietary naming schemes from the naming schemes used by other vendors and security guide authors to avoid any possible challenges regarding copyright violations. Similar to challenges faced by configuration guide authors, the vendor must also choose the appropriate level of abstraction or granularity for their controls or checks so that the units of information in their tool are best aligned not only with the end user's immediate needs, but also so that the tool integrates well with the other tools, capabilities, and stakeholders involved in the configuration management lifecycle at the user's organization.

Lacking CCE-IDs in both the list of desired configuration controls that forms the input to the process and in the configuration management tool, the user must rely on detailed human analysis to successfully map the configuration control list into the tool. To accomplish this, the user will need to rely on her analysis of detailed documentation provided by the vendor to each control. The mapping errors produced by this process are similar to the mapping errors faced by the system designers. A control in the tool that matches a control in the input list might not be recognized as being the same. Conversely, the administrator may map an input control to a control in the tool when, in fact, the controls are different. Worse, if the controls in the tools and the controls in the input list (and the rest of the organization's configuration management practices) are at different levels of abstraction, it will be practically impossible to map the input list into the tool with a high degree of one-to-one correspondences.

4.3.3 How CCE Helps

When both the input list of controls (generated by another part of the organization's internal configuration management practice) and the controls in the vendor's product are both correctly annotated with CCE-IDs, the administrator can quickly and accurately correlate the data sets with each other, which results in lower deployment costs and a minimization of errors which could otherwise be propagated throughout the rest of the organization's configuration management lifecycle. More fundamentally, if both the tool and organization's manner of managing configuration controls are both at the same level of abstraction, then the tool's results can be integrated into the organization's configuration lifecycle with a maximum number of one-to-one correspondences.

It is worth noting at this point that the majority of controls in configuration management and assessment products are managed on a "per platform" basis that is well aligned with CCE's

platform groups. By treating similar controls from different platform groups as being distinct, CCE-IDs are more closely aligned with existing configuration management tools and with the internal practices at organizations that produce internal configuration standards.

In this scenario, CCE-IDs act as boundary objects across the boundaries that exist between the configuration management tool and the customer's organization. To be successful with their individual mandates, both the vendor and customer's organization must forge and maintain a durable shared practice with respect to recognizing and managing configuration controls. By providing a sharable set of identifiers and by articulating a sharable level of abstraction for CCE definitions, CCE helps these different groups coordinate their practices.

4.4 Audit Tool Result Integration

Like the previous use case, this use case can be seen as a part of the configuration management lifecycle. However, this use case typically is only associated with large enterprise organizations that need to integrate the results from multiple configuration audit tools deployed by different parts of their organization. This use case may be properly seen as involving coordination and communication between multiple configuration audit tool vendors, as proxied by the organization attempting to integrate the results.

4.4.1 Scenario

An internal developer at an organization is responsible for consolidating system audit reports from several different sources in order to create and support an integrated reporting capability. The data sources he must integrate range from manually conducted internal system audits, manually conducted external third-party audits, and the results from several commercial audit tools that are deployed by multiple parts of the enterprise. The intended uses for the integrated reporting capability might include situational awareness for IT operations, decision support for IT management, or compliance reporting.

To fully accomplish this task, the developer must do two things. First, he must create a representation of configuration controls that can capture the maximum number of configuration controls tested for by the various assessment capabilities. Second, he must map each of different assessment capabilities into this centralized representation.

4.4.2 Problem

Lacking CCE-IDs, the difficulty lies in the fact that the findings from each data source express the results using proprietary terminology. The manual process of integrating the various audit reports is extremely time consuming and error prone in the same way that other mapping exercises are. Comparisons must be done based on crude searching mechanisms such as keyword searches and must be verified by labor intensive analysis of the individual items. The common error modes are similar as well. Items from different tools that check for the same configuration controls are not identified as being the same and items are matched with each other when, in fact, they check for different issues. More fundamentally, if and when the different tools, manual audit techniques and the internal configuration management practice all use different levels of abstraction to represent configuration information, there are instances where one-to-one correspondence between items is impossible to achieve. As a result, integrated reporting is

frustrated by extensive labor costs, costly errors, and ambiguous results that are a consequence of level of abstraction mismatches.

4.4.3 How CCE Helps

If all of the configuration tools and capabilities are tagged with CCE-IDs, the developer can map them together faster and with more accuracy. More deeply, if the developer builds his internal, unified representation of configuration issues based on CCE, then it is more likely that his reporting capability and the tools will all be managing and representing configuration information at roughly the same level of abstraction, thereby maximizing the number of instances in which data is correlated with a 1-1 correspondence.

It should be noted that the majority of configuration assessment checks in commercial products are created and distributed on a “per platform” basis that corresponds closely with CCE’s platform groups. By assigning different CCE-IDs to similar issues that appear within different versions of platforms, CCE-IDs are in closer alignment with the common practices encoded in the auditing tools. This ensures a maximum number of one-to-one correspondences between CCE-IDs and checks with assessment products.

(We should also mention that it is common for configuration audit capabilities and tools to test for configuration issues for which there are currently no CCE-IDs. In particular, it is common for audit tools to contain tests that produce list of system objects so that an administrator can review the list and make decisions as to whether or not the list is appropriate. For example, a common check of this type is to produce a list of all local users that have administrative rights on the system. The list of users with administrative privileges must be manually reviewed before an audit finding can be created. As of the time of writing of this document, CCE-IDs have not been issued for these types of issues, although they may at some point in the future.)

In this scenario, CCE-IDs act as boundary objects across the boundaries that exist between the different configuration assessment tools and capabilities. While the specific tools vendors may or may not intend to coordinate and communicate with each other, the data integration needs of the organization demand that the tool output be integrated. In a sense, the organization’s developer who is responsible for the data integration acts as a proxy through which the vendor tools communicate with each other and the specific work by each proprietary tool is coordinated with the work done by other tools. By providing a sharable set of identifiers and by articulating a sharable level of abstraction for CCE definitions, CCE helps the proprietary practices within the different tools be coordinated with each other.

4.5 Regulatory Compliance

This use case involves the cross-boundary coordination and communication that takes place between practitioners within the configuration management lifecycle and stakeholders with regulatory compliance interests. These stakeholders may be internal to the organization or external to the organization, but in either case, they are not directly involved in the configuration management lifecycle directly. Rather, they are interested in how the organization’s configuration management practices support certain regulatory compliance goals.

These regulatory goals may be motivated out of a need to comply with applicable laws and accords such as Sarbanes-Oxley, HIPAA, FISMA, Basel II, industry partnership standards such as VISA/PCI, or even their own internal policies. However, IT controls are not mapped into laws or regulatory mandates directly. Rather, compliance is judged from the perspective of security standards or auditing frameworks. Examples of frameworks include ISO 17799, COBIT, NIST sp800-53, ITIL, and DoD 8500.2.

4.5.1 Scenario

A member of an organization's configuration management team has been tasked with documenting how their internal configuration standard for a particular platform comply with or support a particular set of regulatory compliance requirements. The deliverable that she is responsible for providing is a document that maps the configuration standard into a set of requirements, which is often derived from a framework document. At the same time, an auditor is tasked with reviewing the organization's configuration standards to determine how well it complies with or supports the list of regulatory goals.

The collaboration between the configuration management team and the auditor will take on one or both of two forms. First, the auditor may choose to directly inspect artifacts and documents from the configuration management practice that represent or document how the configuration standard is defined and deployed. This might include the written configuration standard produced by the system designer, and shared with the rest of the configuration management team. This might also include inspecting data stored in the configuration deployment tools being used such as active directory group policy objects. Or this might include review of configuration audit tool results.

The second way that configuration management team and the auditor may communicate is via a report that maps the specific configuration controls to the list of regulatory requirements. This mapping is sometimes referred to as a requirements traceability matrix (RTM). Since framework documents are written to address a different point of view that is typically platform independent, it is common for RTMs to contain both one-to-many and many-to-many relationships.

4.5.2 Problem

Without CCE-IDs, the configuration management team must present their configuration standard for the platform using either their own internal naming conventions or by borrowing from a non-standardized naming convention from one of their many tool sets. While their internal naming choices may have some degree of familiarity to those people working within the internal configuration management lifecycle, it will not be familiar to the auditor. So, without CCE-IDs the auditor and configuration management team must work to make each individual configuration item comprehensible to the auditor. To accomplish this, they will need to rely entirely on the configuration team's internal documentation to help clarify each configuration item and then map that to the auditor's own expectations of the list of controls. As with other mapping exercises, this process is labor-intensive and error-prone.

More deeply, it is not uncommon for auditors to think about configuration controls at different levels of abstraction in different contexts. It is not uncommon for auditors and practitioners

within the configuration management lifecycle to think about and discuss configuration controls at different levels of abstraction.

4.5.3 How CCE Helps

When the organization annotates its own internal configuration standards with CCE-IDs, it becomes easier for the auditor to assess the compliance. With CCE-IDs, it would be possible for the auditor and configuration management team to quickly and accurately compare the organizations RTM with other industry best practice RTMs. That is, by standardizing the identification of the configuration controls with CCE-IDs, it becomes possible to articulate standardized expected RTMs for a particular platform. Also, with CCE-IDs the auditor can quickly and accurately review how the configuration standard is implemented across all aspects of the configuration management lifecycle.

In this scenario, CCE-IDs act as boundary objects between auditors and participants in the configuration management lifecycle. In particular, we note that by establishing a standard at the level of abstraction that is common among practitioners within the configuration management practice and codifying that level of abstraction within CCE, CCE-IDs help clarify the otherwise mutable concept of a configuration control for the auditor, which allows the auditors to better orient themselves to the expressions of controls used by the configuration management practitioners. We must emphasize that the on-going practice of cooperation between auditors and configuration managers is durable enough that it is possible for auditors to ultimately comprehend the controls, despite their own inclinations to think of controls at different levels of abstraction. However, the CCE-IDs acting as boundary objects help facilitate the mutual orientation of the auditing and configuration management practices to each other.

5 Definition of CCE

5.1 CCE Core Concepts

The following is a high-level overview of the core concepts in the CCE creation process.

5.1.1 References and Technical Mechanisms

CCE Entries can be thought of as being formed by a set of comparable “references” about configurations. These references are of two different kinds. The first kind consists of objects or constructs from within the platform itself. We refer to these as technical mechanisms. Examples include files, users, registry keys, local security policy settings, and graphical user interface (GUI) controls.

The second kind of configuration reference consists of statements embedded in “resources” that describe the platform, where the word “resource” is taken to include a very broad category that includes:

- Prose security guides that contain natural language configuration statements (e.g., NSA security guides, CIS Benchmarks, DISA STIGS).
- Online help resources that contain natural language configuration statements but may be provisioned using structured data (e.g., MS TechNet, UNIX main pages).

- Structured guidance documents that contain structured configuration statements (e.g., XCCDF checklists).
- Configuration audit or management tools that contain checks, control items or GUI controls.

We refer to the individual statements within a resource as references. We collectively refer to both references (from a resource) and technical mechanisms (with the platform) as references.

5.1.2 Comparable References

If we consider the set of references for a platform, we recognize that some are essentially the same despite differences in their expression. We refer to these as "comparable references" and briefly describe kinds of comparability to illustrate the concept.

- Parameters: In Microsoft Windows XP, you can establish a minimum password length that will be enforced for all newly created user accounts. We consider a citation that specifies a minimum password length of eight and another citation that specifies a minimum password length of 14 as being comparable. Both references are about the same configuration affect or issue. They differ only in their specific value.

- Multiple Mechanisms: In Microsoft Windows XP, you can establish the minimum password length by using a registry editing tool to create and set a specific registry key. Or you can accomplish the goal by using the local security policy tool. Or you can use an active directory group policy editor tool to achieve the same goal. We would consider these three technical mechanisms to be comparable. While these different technical mechanisms may have subtle but significant differences and implications, they are comparable in that they are about the same configuration affect or issue. They differ only in the means used to achieve that affect.

Achieving consistency in the decisions about what constitutes comparable (or non-comparable) references is, in practice, the most difficult aspect of the analytics of CCE creation. The current state of the art of these editorial decisions is documented in the "CCE Editorial Policies" document posted on the CCE Web site, a living document that reflects the evolving community understanding of comparability of references.

5.1.3 Reference Clustering and CCE Creation

The following analogy is intended to explain how comparable references form a CCE entry.

If we were to plot the set of references and technical mechanisms on the plane while grouping comparable references near each other, the result might look something like constellations of stars. See Figure 1.

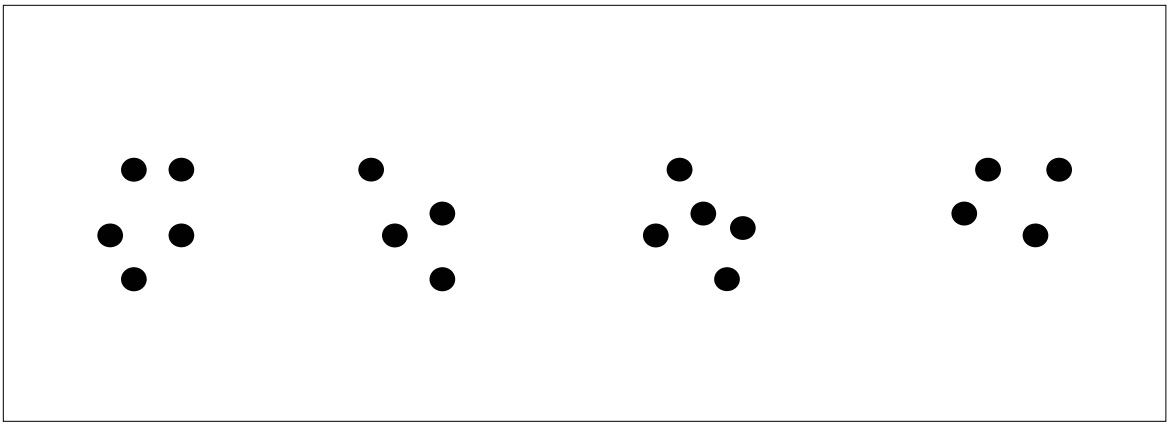

Figure 1. Clusters of comparable references plotted on the plane.

In this representation, we would say that a cluster of comparable references forms or creates a CCE entry. Stating it another way, a CCE entry is an identifier that is associated with a set or cluster of comparable references. See Figure 2.

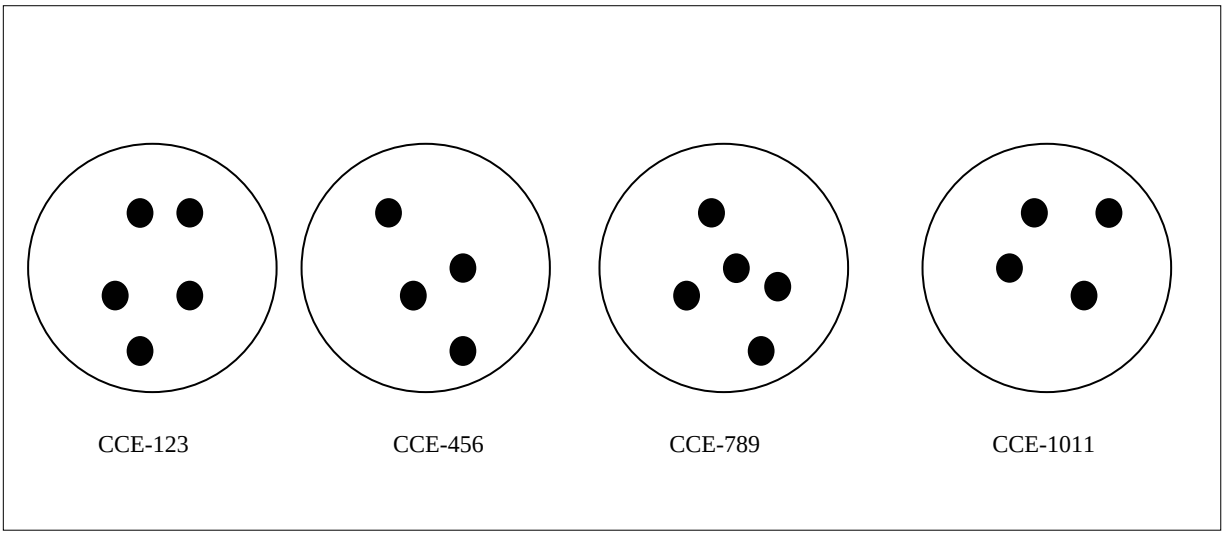

Figure 2: CCE-IDs assigned to clusters of comparable references.

The core problem involved in CCE creation is that given a set of references and technical mechanisms about a new platform as input we:

1. Identify clusters of comparable references.
2. For each cluster output a new CCE-ID.

The core problem involved in CCE maintenance is that given a set of references and technical mechanisms about a platform for which CCEs currently exist we:

1. For each new reference search the CCE data to determine if there are any comparable references.

2. If comparable references are found, merge the new reference into the existing CCE.
3. If no comparable references are found, create a new CCE.
4. Output an updated CCE List.

5.2 Conceptual Data Model

Having discussed how we come to know of a CCE, we are now able to discuss how to represent CCE data. This section will describe the CCE data model at a conceptual level. There will be many implementation and data representation issues that will not be discussed in detail here.

The CCE data model consists of five (conceptual) entities:

- Externally defined platform object schemas (e.g., OVAL, WMI, WBEM).
- Technical mechanisms (expressed in terms of OVAL, WMI, or similar).
- Resources (e.g., prose security guides, XCCDF documents, audit tools).
- References (line items within a resource such as XCCDF rules or audit checks).
- CCE entries (composed of one or more references and/or technical mechanisms).

We now discuss each of these entities in more detail.

5.2.1 Externally Defined Platform Object Schemas

The CCE data model will depend on a set of schemas that are defined and maintained outside of the CCE project. CCE needs a way to refer to platform objects such as specific registry keys, files, users or elements within configuration files. To do this, CCE needs a way to represent these platform objects. Externally defined platform object schemas provide CCE with this capability. Examples of externally defined schemas might include: MITRE's OVAL, WMI, and WBEM.

We should make three observations about the use of externally defined objects. First, CCE will adopt the convention used by XCCDF in its use of externally defined schema for XCCDF checks. XCCDF rules can be associated with checks. These checks are represented according to some schema that is defined outside of XCCDF. XCCDF rule references to checks contain two bits of information then: a reference to the check schema (e.g., MITRE's OVAL) and a reference to a specific check defined according to that schema (e.g., a single OVAL-ID). In the same way, we will allow technical mechanisms to be described by multiple platform object schemas.

Second, while we want to facilitate the use of different schemas to describe platform objects, we recognize that not all schemas will provide sufficient richness. For this reasons, we anticipate the need to exclude certain schemas for certain platforms while approving others.

Lastly, while CCE will strive to capture technical mechanisms to platform objects with structure, we recognize that the CCE project is positioned to serve as a transitional step from natural language to formalized data structures. For this reason, we will need to retain the ability to fall back to a default "plain text" representation of technical mechanisms as a recognized externally defined schema.

5.2.2 Technical Mechanisms

Technical mechanisms are representations of specific objects within a platform. Examples include specific:

- Registry keys.
- Files.
- Directories.
- Users.
- User groups.
- System or network services.
- Configuration file entries.
- GUI controls (e.g., local security policy or GPO paths).

We should make two observations about technical mechanisms. First, technical mechanisms must be associated with at least one platform. For example, we can refer to a specific registry key within the Windows XP operating system. This same registry key may also be used in Windows Vista. But, it does not make sense to discuss a technical mechanism that is not associated with at least one platform.

Second, technical mechanisms are not associated with a particular setting or configuration choice. For example, a technical mechanism might refer to a specific registry key in Windows XP, but it would not be associated with any specific value for that registry key. Any statement that would be associated with specific values for any technical mechanism will be represented as a citation (defined below).

Technical mechanisms will have the following attributes:

- ID: Unique identifier for technical mechanism.
- Coverage: CPE identifier for the platforms covered by the technical mechanism.
- Mechanism-Type: Namespace of the schema for an underlying technical mechanism (e.g., OVAL object, Apple plist, WMI).
- Mechanism: Contents of the technical mechanism expressed in the aforementioned namespace and schema.

5.2.3 Resources

Resources are the things from which individual references are drawn. Examples of resources include:

- Prose security guides that contain natural language configuration statements (e.g., NSA security guides, CIS Benchmarks, DISA STIGS).
- Online help resources that contain natural language configuration statements but may be provisioned using structured data (e.g., MS TechNet, UNIX man pages).
- Structured guidance documents that contain structured configuration statements (e.g., XCCDF checklists).
- Configuration audit or management tools that contain checks, control items or GUI controls.

The attributes that describe a resource are based on the Dublin Core[13] set of attributes to describe a resource and include:

- ID: Unique identifier for configuration document.
- Resource: Name of document (e.g., Filename, URL).
- Publisher: Name of publishing organization (e.g., NIST, CIS, Microsoft).
- Date: Date of publication.
- Title: Title of configuration document.
- Version: Version number of the configuration document.
- Format: Format in which configuration document is expressed (i.e., XCCDF, OVAL, document, Web page, software product).
- Coverage: CPE Identifier for the platforms covered by the resource.

5.2.4 References

References are individual statements about configurations that are drawn from a resource. References are always associated with the resource from which they are drawn. References differ from technical mechanisms in two ways. First, references are statements made about the platform whereas technical mechanisms exist within the platform. Secondly, a citation is typically associated with suggested value for a configuration setting, whereas a related technical mechanism describes the mechanism by which the configuration is enforced.

For example, a citation might say that the FTP service should be disabled. An associated technical mechanism might discuss the Local Security Policy navigation path in Windows XP that allows the user to make this setting. The local security policy navigation path does not make any assertions about how the FTP service should be configured.

The attributes that describe a citation include:

- ID: Unique identifier for CCE citation.
- Resource: Many-to-one relation with a CCE source.
- Citation-Type: Namespace of the schema for a recommendation line item.
- Citation: Contents of the recommendation expressed in the aforementioned style.

5.2.5 CCE Entries

CCE Entries are the core elements in the CCE data model. They consist of two kinds of information. The first are the descriptive elements that describe the CCE in human understandable manner. Primarily, this consists of a textual description of the configuration element (with no specific recommendation) and the logical parameter values that might be associated with the control. For example, logical parameters might include “enabled” or “disabled”, not the actual storage values associated with these conceptual values.

The other kind of information in the CCE element are the links to associated references (prose statements, XCCDF, audit checks) and technical mechanisms (files, users, registry keys, permission bits).

[13] http://dublincore.org/documents/1999/07/02/dces/

The attributes of the CCE entry include:

- ID: Unique identifier for Common Configuration Enumeration (CCE-####-#)
- Definition: Description of configuration control
- Logical Parameters: Possible conceptual values
- Citation: One-to-many relation with references
- Technical-Reference: Zero-to-many relation with technical mechanisms

5.3 Current CCE Creation in Practice

Having discussed the basic data model, we should discuss the flow of content creation that has occurred during the past two years. Typically, an analyst first encounters a configuration statement in one of two ways.

The first and most common way is she encounters a configuration guidance statement such as "The minimum password length should be set to 8." This statement would be found in a resource (document, audit tool) the analyst is using or creating. This would lead to the identification of the resource itself (resource entity). It would also lead to the capture of the configuration statement as a citation (citation entity). The next step for the analyst is often to determine how to effectively set the configuration or test for its presence. In this example, this would lead her to various registry keys, local security policy settings or GPO settings, each of which would be captured as technical mechanisms (technical mechanism entity). After the configuration statement and the associated technical mechanisms are identified, the analyst could then generalize the configuration issue and capture that as a CCE description and conceptual parameters (CCE entity). In this way, the CCE entry binds together the references and technical mechanisms.

The second way an analyst creates a CCE entry is when she begins with a particular configuration control such as a particular registry key, file, or GPO setting. In this context, the analyst is considering the control without any specific external guidance on how the control should be set. Instead, she may be working to formulate what her own suggestion would be. Starting with single technical mechanism (e.g., a registry key), she then typically expands her scope to consider what other technical mechanisms would have an equivalent effect (e.g., local security policy settings, GPO settings). These are captured as further technical mechanisms (technical mechanism entity). Once a set of comparable technical mechanisms are identified, she can generalize the technical mechanisms and create a CCE entry as described above.

We should emphasize that these two methods are how the current set of CCE entries have been created. The first of these, starting with a guidance statement and then working towards the technical mechanisms, is by far the most common. In this way, we recognize that CCE entries are derived from sets of comparable references and technical mechanisms.

6 CCE Creation Process

6.1 Community Cooperation and the CCE Creation Process

The following section discusses a variety of issues related to how the community and different organizations might cooperate on the creation and maintenance of CCE data. The discussion will raise and motivate several technical issues. We will attempt to keep the focus of this section on the inter-organizational issues and defer the discussion of technical implications.

6.1.1 The Basic Content Flow, Starting Assumptions, and Implications

The CCE creation process begins with the creation of a new platform. The end goal of the CCE project is to make structured CCE data available to the world. The U.S. National Vulnerability Database (NVD)[14], hosted by NIST, would be one such provisioning capability. The question then is to define the process that spans these two end-points. See Figure 3.

Figure 3. The end points of the CCE creation process.

With the end points of the process identified, we now make some of the assumptions about the process more explicit. These assumptions will serve as the "guardrails" by which we will bound the discussion of the various process options by which CCE might operate. Needless to say, different assumptions may lead us to different processes.

Assumption 1: No single organization has the subject matter expertise or charter to create CCEs for all platforms of interest.

We recognize that no single organization has the charter, capability, or subject matter expertise to produce CCE data for all platforms of interest. This leads us to the following conclusion.

The CCE process must accommodate the involvement of multiple organizations in order to achieve adequate coverage of all platforms of interest.

Assumption 2: Multiple organizations will (continue to) publish different but overlapping resources for high-interest platforms.

[14] http://nvd.nist.gov/nvd.cfm

The creator of the platform is clearly best positioned to define the set of references and technical mechanisms that define their platform and to create the CCEs for that platform. However, there are other subject matter experts (SME) who are also chartered to make statements about configuration choices for that platform. These include:

- Governmental authors of prose security guides such as NSA and DISA.
- Governmental authors of structured configuration guidance such as NIST's Security Content Automation Program (SCAP)[15].
- Commercial authors of prose security guides such as the Center for Internet Security.
- Creators of information sharing schemas such as WMI, WBEM, CMDB, OMG, and others.

We recognize historically that different platforms will have different subsets of these types of SMEs involved in the creation of content that would feed into the CCE creation and maintenance process. And while many have strived to unify configuration guidance within a particular context, we recognize that for high-interest platforms that are deployed in many different contexts, there will continue to be a wide range of SME published resources that should be taken into consideration in the CCE content creation process. Thus, for a platform (especially high interest platforms) the CCE process must provide some mechanism by which configuration statements made by multiple organizations can be merged together.

Assumption 3: CCE data will be provisioned to the world via the Web.

The full indexical utility of CCE-Identifiers to foster fast and accurate correlation of configuration information will only be realized if and when they are widely used to tag configuration statement through out all aspects of the industry. The World Wide Web provides the most obvious capability for the distribution of CCE data. Currently, the MITRE Corporation is providing this capability. NIST's NVD will also offer a Web-based provisioning capability.

Assumption 4: CCE data will have sufficient completeness and correctness to be useful.

The utility of CCE data will be determined by its basic correctness. As Wikipedia has demonstrated, community feedback and input into the content creation and maintenance process is a powerful mechanism by which content can be provisioned in an "emergent" manner. While the CCE process may attempt to leverage this power, we also note that Wikipedia has demonstrated that malicious subversion of content creation and maintenance processes is a very real and likely prospect, especially when large state and financial interests are at stake.

We also recognize that any federated content creation process is subject to drift and internal inconsistencies, even when all participating parties are well intentioned. Lastly, we recognize that any federated content creation process opens up the possibility that authorized participants might disagree about the formation and correctness of particular data elements.

[15] http://nvd.nist.gov/scap.cfm

Lastly, we recognize that historically the different resources (e.g., guides, XCCDF, schemas, tools) produced by different SMEs about the same platform have different content orientations. And some of these resources fail to provide enough information to adequately create a CCE entry. This implies that a necessary aspect of the information merge process is the extension of some SME authored resources to include all information needed to create a CCE entry. This might mean requiring the authoring SMEs to extend their traditional content creations processes to include the necessary additional information. Or it might mean extending the SME data by those responsible for the information merge. We note that the MITRE Corporation is currently fulfilling this role in the CCE creation process.

We conclude that the CCE process must provide some mechanisms for guaranteeing an adequate level of consistency, completeness and correctness in the CCE data. These controls might include training and documentation for those involved in CCE data creation, controls on who is authorized to create and edit CCE data, some centralized review and oversight capability, copyright, and redistribution controls or all of the above.

Assumption 5: CCE data will have sufficient structure to be useful.

We recognize that for CCE to be adopted and used widely across the industry, it must be provisioned in a manner that allows the data set to be parsed and modified. Currently, MITRE publishes the CCE List as spreadsheets (in MS-Excel format) and plans on releasing a version of the data in XML format.

We recognize that the resources published by platform SMEs range from prose documents (e.g., MS-Word, PDF) to structured resources (e.g., XCCDF, WMI, CMDB, OVAL).

We conclude that the CCE process will need to have a defined schema for the shared representation and transport of CCE-related data. We also conclude that the process will need to include a step by which unstructured and structured data is merged into this centralized CCE data format.

We are now in a position to describe the basic elements of the CCE creation process. Note, this description is made at the conceptual level, not the operational level. The actual order of the processes may be subject to change and various steps may be performed by the same organization.

In particular, we make no assertion at this point about which organizations should perform which tasks. We only strive to state the different functional roles that need to be performed in any CCE creation process. See Figure 4.

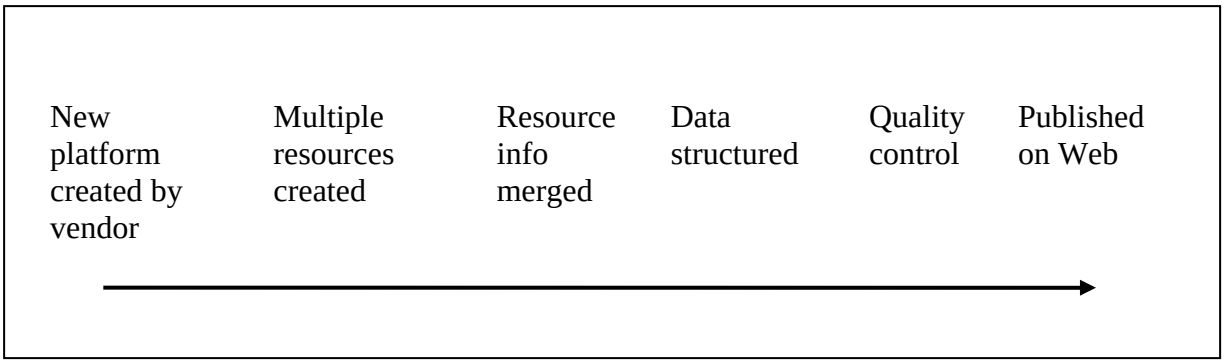

Figure 4. The conceptual steps of the CCE content process.

6.1.2 One More Assumption and CVE Lessons Learned

CCE has grown out of MITRE's CVE project, which offers us some potential lessons learned regarding a federated model of CCE content creation. CVE maintains a relationship with a set of organizations referred to as Candidate Naming Authorities (CNAs). The list of current CNAs includes Apple, Debian, Microsoft, and Red Hat among others. CNAs are given some amount of training on the proper assignment of CVE Identifiers (CVE-IDs), especially regarding the CVE content decisions. They are then given blocks of CVE-IDs to assign to vulnerabilities within their advisories. In this way, CVE today is operating on a federated CVE assignment model.

CNAs release their newly assigned CVE-IDs to the public in their security advisories. These advisories are typically released to the public as text or HTML publications and they typically vary considerably in terms of the amounts and kinds of technical details that are disclosed.

The creation of a structured CVE data stream is done based on a "pull" model. The CVE content team learns about new CVE-IDs issued by CNAs by way of the advisories. The CVE team then pulls the data down from these advisories, analyzes it for correctness and completeness, and then puts it into a structured format.

The CCE effort is taking place in a different environment in terms of content creation and provisioning. Adoption of semantically rich information standards for expressing configuration guidance such as the NSA's XCCDF standard and for expressing technical details such as MITRE's OVAL may give us the opportunity to improve upon the CVE model to gain efficiencies in the CCE creation process. This leads us to our final assumption.

Assumption 6: The CCE process will leverage structured configuration content standards and their supporting content creation processes to achieve efficiencies in the CCE creation process, thereby minimizing the need for a centralized CCE creation role.

That is, we recognize that (1) the SMEs for a given platform are best positioned to provide the information needed to create CCEs, and (2) these same SMEs are increasingly using information standards like XCCDF and OVAL. We conclude that it would be natural to extend these developing capabilities among the SMEs to also provision structured CCE data. In short, the more structured the data is that is created by the SMEs, the smaller the burden becomes on the other stages of the CCE creation process.

7 Proposed CCE Creation Process

We can identify three variations on how CCE-related information might flow. These three use cases are based on the experience of the MITRE CCE team as it has been coordinating and collaborating with different organizations for the creation of the current CCE data set.

One of the key observations to make is that SMEs most often engage with configuration issues and thus CCEs as a part of a larger effort. For example, the SME may be involved with the creation of a guidance document to cover all secure configuration issues for a platform. Or the SME may be involved with the creation of a battery of configuration checks. In these cases, we note that SME is (1) typically dealing with a larger number of configuration issues at a time, and (2) typically merging their own proprietary knowledge/content with other content (such as CCE) that relates to the platform.

We observe that the most common manner in which this is currently achieved by SMEs today is that they work with a copy of existing CCE data in an off-line mode such as XML or a spreadsheet and then they do bulk processing against that data set. This typically involves the merging of the CCE or CCE related data with their own data sets and it often involves collaboration with other SMEs within their organization. Once this merged data set is finalized, it is then given to the MITRE CCE team for further processing and merging with the official CCE data set. We will refer to this basic process as “bulk” processing.

7.1 Bulk New Create

A bulk new create takes place whenever a set of CCEs is created for a platform for which no current or related CCEs exist. This describes how the first CCE List was created for Windows 2000 and XP. Based on our involvement with SMEs who are creating guidance for applications, we anticipate this will be the situation for many application oriented CCEs, at least initially.

The primary distinguishing aspect of the bulk new create situation is that the SME has assurance that no existing CCE data has any bearing on the configuration issues they are defining for the platform in question. For this reason, there is no need for a robust search capability into the existing CCE data set while the SME is creating new CCE content.

We propose that the most effective organization to produce CCE content in this situation is the platform’s vendor. The vendor has the best understanding of the conceptual security model (captured in the CCE entity) and the complete set of comparable technical mechanisms used to implement that security model (captured in the technical mechanisms entity). Further, we anticipate that the vendor is best positioned to author the authoritative secure configuration guidance for their platform as a part of their dialogue with their government and industry customers. Assuming that the vendor will commit their own SME to the task of creating authoritative guidance (e.g., in support of the NIST SCAP program), it would be natural and most effective to extend this work slightly to produce CCE data.

Provided that the related CCE data is created correctly and provisioned with the appropriate structure, it could be merged into the larger CCE data set with minimal costs.

7.2 Bulk Update and Merge

As CCE matures and grows in terms of the number of platforms, it will be more common for bulk processing to change. The primary aspect of this use case that separates it from the bulk new create is that the SME must assume that CCE data may already exist that is related to the set of configuration controls they are considering. This is certainly the case when the SME is responsible for mapping their own resource (document, XCCDF, audit tool) into the CCE data set for a platform that has already been covered by CCE. This is also the case when the platform vendor releases a new version of a platform. Typically, new versions carry forward a large percentage of the same security model from prior versions and thus it is common to see CCE span multiple versions of a platform.

We propose that the most effective organization to produce CCE content in such situations are either the resource owner (in the case of mapping a resource into CCE) or the platform vendor (in the case of an updated version of a platform already covered by CCE). In both cases, the basic workflow remains the same. The SME will be engaged with the consideration of a bulk of their own proprietary references or technical mechanisms and will work in an off-line mode to merge their data with a copy or subset of the CCE data. Once the bulk update is composed, it will need to be merged into the current CCE data set. The same output requirements apply. The needs need to be correct and provisioned with the appropriate structure.

Two additional technical capabilities must also be considered to support this use case. The first is a robust search capability. As the SME considers any one of their references or technical mechanisms, they must determine if the CCE data set contains any references or technical mechanisms that are comparable. There are three challenges to be considered:

- The search must be able to adequately deal with references that are largely stored as text strings. This includes text snippets lifted from prose documents, XCCDF documents and description fields from audit checks.

- It must work across multiple representations of platform objects according to different modeling schemas. For example, we will likely need to be able to recognize a specific registry key as being the same regardless of whether it is represented in OVAL or WMI.

- The search must be robust enough to provide the SME with a high assurance that a negative result is valid. That is, if the search result is conceptually equivalent to “not found,” the SME needs to have a high degree of assurance that they should create a new CCE entry as opposed to merging their references into a currently existing CCE.

The second supporting technology that will be needed to support this use case is a federated content management capability that will allow multiple organizations to perform bulk edits of the CCE data set and then have those edits effectively merged.

7.3 Individual Creates and Edits

The third and final use case is that of individual edits to the CCE List. SMEs peruse the CCE data set for a variety of reasons. It is not uncommon for SMEs to detect things that need to be added or corrected in the data set. For this reason, we want to make it possible for trusted CCE users to be able edit and update existing elements or add missing ones.

8 Additional Information

The most current information about CCE and the latest version of the CCE List are available on the CCE Web site at http://cce.mitre.org.

Copyright © 2008, The MITRE Corporation. All rights reserved. Permission is granted to redistribute this document if this paragraph is not removed. This document is subject to change without notice.