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

NIST Interagency Report 7502
(Draft)

The Common Configuration Scoring System (CCSS) (DRAFT)

Karen Scarfone
Peter Mell

NIST Interagency Report 7502 (Draft)

The Common Configuration Scoring System (CCSS) (DRAFT)

Karen Scarfone
Peter Mell

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

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

May 2008

U.S. Department of Commerce

Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology

James M. Turner, Deputy Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 7502 (DRAFT)
24 pages (May 2008)

Acknowledgements

Portions of this report are based on the official Common Vulnerability Scoring System (CVSS) standard[1] from the Forum for Incident Response and Security Teams (FIRST) CVSS Special Interest Group and on NIST Interagency Report (IR) 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*[2]. The authors sincerely appreciate the contributions of CVSS Special Interest Group members and others in reviewing drafts of this report, including Kurt Dillard, Robert Fritz, Tim Grance, Ron Gula, Dave Mann, Doug Noakes, Jim Ronayne, Murugiah Souppaya, and Kim Watson. Special thanks also go to Chuck Wergin and Dan Walsh, analysts for the National Vulnerability Database (NVD), for testing the proposed scoring system by scoring many configuration entries with it, as well as reviewing several drafts of this report.

Abstract

The Common Configuration Scoring System (CCSS) is a set of standardized measures for the characteristics and impacts of software security configuration issues. CCSS can assist organizations in making sound decisions as to how security issues should be addressed and can provide data to be used in quantitative assessments of the overall security posture of a host. This report defines proposed measures for CCSS and a formula to be used to combine the measures into scores for each configuration issue. The report also provides several examples of how CCSS measures and scores would be determined for a diverse set of configuration issues.

1 http://www.first.org/cvss/cvss-guide.html
2 http://csrc.nist.gov/publications/PubsNISTIRs.html

Table of Contents

List of Tables

1. Introduction

Operating systems (OS) and many applications have one or more configuration options available that affect their security. For example, an operating system might offer access control lists that can be used to set the privileges that users have for files, and an application might offer a setting that can enable or disable a feature that encrypts sensitive data stored by the application. These settings are known as *security configuration settings*. The security of an OS or application may also be altered by means other than settings, such as uninstalling software features. An element of a software's security that can be altered through the software itself (i.e., by security configuration settings or other means) is referred to as a *security configuration issue*.

Each security configuration decision can have positive and negative effects of varying degrees to the security of a host. Without a standardized way to quantify these effects, organizations cannot easily make sound decisions as to how each security issue should be addressed, nor can they quantitatively determine the overall security strength or weakness for a host. Being able to express the major security characteristics of configuration issues through standardized measures also assists organizations in decision-making, particularly when considering how security controls such as firewalls might mitigate threats against certain issues. For example, if a configuration causes a weakness that an attacker could exploit across networks, then a network firewall, intrusion detection system, or other network-based security control might be able to lessen the risk of exploitation of that weakness.

This report proposes a set of measures for security configuration issues and a formula to combine the measures into scores for each issue. The definitions for these measures and the formula are collectively called the Common Configuration Scoring System (CCSS). CCSS is derived from the Common Vulnerability Scoring System (CVSS), which is designed for use in measuring the relative severity of vulnerabilities caused by security-related software flaws. CCSS uses the basic components of CVSS and adjusts them to take into account the differences in software flaws and security configuration issues.

At this time, CCSS only addresses the intrinsic characteristics of security configuration issues that are constant over time and environments. The characteristics addressed by CCSS involve how readily a weakness can be exploited and how exploitation can impact hosts. These characteristics are known collectively as the *base metrics*, and they are the inputs into the equation that calculates the *base score*. For the base metrics, other factors that might affect weaknesses, such as the presence of additional security controls, the configuration of other settings, or other environment-specific elements, are disregarded. In the future, CCSS will be expanded to include *environmental metrics*, which represent characteristics that are relevant and unique to a particular environment.[3]

By themselves, CCSS base scores can be used as an aid in evaluating the potential relative severity of individual configuration issues. However, CCSS base scores do not take into account mitigating security controls that prevent exploitation or any other organization or environment-specific considerations, so their value as standalone measures is rather limited. When CCSS is expanded to include environmental metrics, organizations will be able to use the metrics as inputs for threat and vulnerability modeling, quantitative risk assessment processes, and other security management purposes. For example, an organization could compare a host's configuration to a baseline configuration and, as part of that analysis, use the CCSS scores for the settings that deviate from the baseline configuration in determining how significantly the host's security deviates from the expected state.

[3] In addition to base and environmental metrics, CVSS also includes a third type of metrics, *temporal metrics*, which represent characteristics that change over time but not between different environments. The temporal metrics in CVSS address whether a vulnerability has been confirmed to exist and has been demonstrated to be exploitable, as well as what level of remediation (e.g., patching) is currently available. These metrics are applicable to software flaws but not to configuration issues, so as of this writing it is not expected that CCSS will have temporal metrics.

Because base scores should be identical across all organizations, the intention is for base scores for configuration issues to be produced by a relatively small number of entities, such as security vendors and organizations that maintain vulnerability databases. Organizations that are end users of base scores should not need to calculate base scores, and given the complexity and difficulty of performing base scoring, it is anticipated that few organizations will choose to expend the time and resources needed to perform their own base scoring. However, environmental scoring is organization-specific, so it will need to be performed by individual organizations.

The primary purpose of this document is to define CCSS, and not to explain in detail how organizations can use CCSS. This document is an early step in a long-term effort to provide standardized data sources and corresponding methodologies for conducting quantitative risk assessments of host security. Additional information will be published in the future regarding CCSS and how organizations will be able to take advantage of it. Currently, the focus is on reaching consensus on the definition of CCSS and encouraging security vendors and other organizations to consider adopting CCSS.

2. CCSS Base Metrics

This section defines the base metrics that comprise the proposed CCSS standard and the two logical groupings of these metrics: exploitability and impact. This section also describes the standard format for documenting CCSS metrics and defines the equation used to generate CCSS scores.

2.1 Exploitability Metrics

The three base metrics for exploitability—Access Vector, Access Complexity, and Authentication—capture how easily the weakness caused by a configuration issue could be accessed and whether extra conditions are required to take advantage of it.

Weaknesses caused by configuration issues can be taken advantage of in two ways: actively and passively.[4] Some weaknesses can be actively exploited, such as an attacker gaining access to a sensitive file because the targeted host is incorrectly configured to permit any user to read the file. Other weaknesses passively prevent authorized actions from occurring, such as not permitting a system service or daemon to run or not generating audit log records for security events. For some of the exploitability base metrics, scoring considerations are dependent on these distinctions, and all such considerations are noted below.

2.1.1 Access Vector (AV)

The Access Vector metric is assigned differently depending upon whether the configuration issue can be taken advantage of actively or passively.

For active exploitation, the metric reflects from where the exploitation can be performed. The score increases with the degree to which an exploiter[5] may be remote from the affected host.

For passive exploitation, this metric reflects from where users or other hosts are supposed to be able to perform the action that is being prevented from occurring. The more remote the users or other hosts can be, the greater the score.

The possible values for this metric are listed in Table 1.

[4] It is theoretically possible that a single weakness caused by a configuration issue could be taken advantage of both actively and passively, but no examples have been found in the limited review conducted to date.

[5] The term "exploiter" refers to a party that is taking advantage of a weakness caused by a security configuration. In some cases, authorized users will inadvertently exploit weaknesses, often without any knowledge of doing so, so terms such as "attacker" are avoided in this document except where the context clearly indicates a conscious attack.

Table 1. Access Vector Scoring Evaluation

Metric Value	Description
Local (L)	Active Exploitation: A weakness actively exploitable with only local access requires the exploiter to have either physical access to the host or a local (shell) account. Examples of locally exploitable configuration issues are excess privileges assigned to locally accessible user or service accounts, directories, files, and registry keys; prohibited local services enabled; weak password policies for local accounts; lack of required password protection for screen savers; and lack of required peripheral usage restrictions, such as permissions for the use of USB flash drives or CDs. Passive Exploitation: A weakness passively preventing actions from occurring with local access affects only local users, processes, services, etc. Examples of these configuration issues are insufficient privileges assigned to locally accessible user or service accounts, directories, files, and registry keys; necessary local services disabled; and overly restrictive peripheral configurations, such as preventing the use of USB flash drives or CDs when an organization's policy permits such use.
Adjacent Network (A)	Active Exploitation: A weakness actively exploitable with adjacent network access requires the exploiter to have access to either the broadcast or collision domain of the software. Local networks include local IP subnet, Bluetooth, IEEE 802.11, and local Ethernet segment. An example of an adjacent network configuration issue is configuring a wireless LAN network interface card to connect to any available wireless LAN automatically. Passive Exploitation: A weakness passively preventing actions from occurring with adjacent network access affects users or other hosts on the broadcast or collision domain of the host. An example of such a configuration issue is a host that is intended to share its Internet access with other hosts on the same subnet, but is configured so that it cannot provide Internet access to them.
Network (N)	Active Exploitation: A weakness actively exploitable with network access means the exploiter does not require local network access or local access. The software with the weakness is bound to the network stack; this is also termed "remotely exploitable". An example is a configuration setting for a network service such as FTP, HTTP, or SMTP (e.g., excess privileges, weak password policy). Another example is a prohibited network service being enabled. Passive Exploitation: A weakness passively preventing actions from occurring with network access affects users or hosts outside the broadcast or collision domain. An example is a host that is intended to provide a network service to all other hosts, but the service has inadvertently been disabled. Another example is a host that interacts with remote hosts, but has all of its logging and auditing capabilities disabled; the host will fail to record events involving the remote hosts, so this could permit malicious activity by those remote hosts to go unnoticed.

2.1.2 Access Complexity (AC)

The Access Complexity metric is assigned differently depending upon whether the configuration issue can be taken advantage of actively or passively.

For active exploitation, this metric measures the complexity of the actions required to exploit the weakness once an exploiter has gained access to the target host. For example, consider a default user account and password for a network service: once the target host is located, the exploiter can access the network service at will. Other weaknesses, however, may require additional steps in order to be exploited. For example, a weakness in an email client is only exploited after the user downloads and opens a tainted attachment. The lower the required complexity, the higher the score.

For passive exploitation, this metric is set to the lowest complexity because the outcome of the weakness, such as not permitting a service to run, has already occurred or is constantly occurring—no additional actions are needed.

The possible values for this metric are listed in Table 2.

Table 2. Access Complexity Scoring Evaluation

Metric Value	Description
High (H)	Active Exploitation: Specialized access conditions exist for active exploitation. For example: • The weakness makes it only slightly easier for an attack to succeed. For example, a weak password length requirement would improve an attacker's chances of guessing or cracking a password, but since each weakness needs to be considered on its own (i.e., we can't assume that the attacker has unlimited opportunities to guess), this particular weakness would not significantly reduce the complexity of attack. • The attacking party must already have elevated privileges (for example, a weakness that only administrators could take advantage of). • The weakness is seen very rarely in practice, so parties that could potentially exploit it are very unlikely to be looking for it. Passive Exploitation: Not applicable to this value.
Medium (M)	Active Exploitation: The access conditions for active exploitation are somewhat specialized; the following are examples: • The attacking party is limited to a group of hosts or users at some level of authorization, possibly untrusted. • Some information must be gathered before a successful attack can be launched. • The weakness is non-default, and is not commonly encountered. • Successful exploitation requires the victim to perform some action such as visiting a hostile web site or opening an email attachment. Passive Exploitation: Not applicable to this value.
Low (L)	Active Exploitation: Specialized access conditions for active exploitation or extenuating circumstances do not exist. The following are examples: • The affected product typically provides access to a wide range of hosts and users, possibly anonymous and untrusted (e.g., Internet-facing web or mail server). • The weakness is default or ubiquitous. • The attack can be performed manually and requires little skill or additional information gathering. Passive Exploitation: Weaknesses that passively prevent actions from occurring or otherwise do not require any actions to be performed are scored as Low. Examples are an audit service that is configured such that it fails to record security events, and an update service that is configured such that it fails to download security updates.

2.1.3 Authentication (Au)

This metric measures the number of times an exploiter must authenticate to a target in order to exploit a weakness. This metric does not gauge the strength or complexity of the authentication process, only that an exploiter is required to provide credentials before an exploit may occur. The possible values for this metric are listed in Table 3. The fewer authentication instances that are required, the higher the score.

It is important to note that the Authentication metric is different from Access Vector. Here, authentication requirements are considered once the host has already been accessed. Specifically, for locally exploitable weaknesses, this metric should only be set to "single" or "multiple" if authentication is needed beyond what is required to log into the host. An example of a locally exploitable weakness that requires

authentication is one affecting a database engine listening on a Unix domain socket (or some other non-network interface). If the user must authenticate as a valid database user in order to exploit the weakness, then this metric should be set to "single."

There are no distinctions between active and passive exploitation in assigning values to this metric.

Table 3. Authentication Scoring Evaluation

Metric Value	Description
Multiple (M)	Exploiting the weakness requires that the exploiter authenticate two or more times, even if the same credentials are used each time. An example is an exploiter authenticating to an operating system in addition to providing credentials to access an application on that host.
Single (S)	One instance of authentication is required to access and exploit the weakness.
None (N)	Authentication is not required to access and exploit the weakness.

The metric should be applied based on the authentication the exploiter requires before launching an attack. For example, if a mail server can be exploited by a command that can be issued before a user authenticates, the weakness should be scored as "None" because the exploiter can launch the exploit before credentials are required. If the mail server can only be exploited after successful authentication, then the weakness should be scored as "Single" or "Multiple," depending on how many instances of authentication must occur before issuing the command.

2.2 Impact Metrics

The three base metrics related to impact, Confidentiality Impact, Integrity Impact, and Availability Impact, measure how exploitation of a weakness caused by a configuration issue could directly affect a targeted host. The impacts are independently defined as the degree of loss of confidentiality, integrity, and availability. For example, a weakness could be exploited to cause a partial loss of integrity and availability, but no loss of confidentiality.

2.2.1 Confidentiality Impact (C)

This metric measures the potential impact on confidentiality of a weakness. Confidentiality refers to limiting information access and disclosure and host access to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. The possible values for this metric are listed in Table 4. Increased confidentiality impact increases the score.

Table 4. Confidentiality Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the confidentiality of the host.
Partial (P)	There is considerable informational disclosure. Access to some host files is possible, but the exploiter does not have control over what is obtained, or the scope of the loss is constrained. Another form of partial confidentiality impact is the use of weak password policies, which make it easier to guess or crack passwords. AND/OR There is considerable, but not total, unauthorized access to the host. Examples include an authorized user gaining access to certain prohibited system functions, an unauthorized user gaining access to a network service offered by the host, and an unauthorized user gaining user or application-level privileges on the host (such as a database administration account).
Complete (C)	There is total information disclosure, resulting in all host files being revealed. The exploiter is able to read all of the host's data (memory, files, etc.) An example is someone who is not authorized to act as a system administrator gaining full administrator privileges to the host.

2.2.2 Integrity Impact (I)

This metric measures the potential impact to integrity of a weakness. Integrity refers to the trustworthiness and guaranteed veracity of information. The possible values for this metric are listed in Table 5. Increased integrity impact increases the score.

Table 5. Integrity Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the integrity of the host.
Partial (P)	Modification of some host files or information is possible, but the exploiter does not have control over what can be modified, or the scope of what the exploiter can affect is limited. For example, OS or application files may be overwritten or modified, but either the exploiter has no control over which files are affected or the exploiter can modify files within only a limited context or scope, such as for a particular user or application account. Another example is poorly configured auditing or logging, which reduces the number of events that are logged or causes the records to be retained for a shorter period of time than needed. AND/OR The weakness can be misused to alter the host's security configuration. An example is a weakness that would allow an unauthorized file (such as one containing malware) to be stored on the host or allow an unauthorized program to be installed on the host.
Complete (C)	There is a total compromise of host integrity. There is a complete loss of protection, resulting in the entire host being compromised. The exploiter is able to modify any files on the target host. An example is someone who is not authorized to act as a system administrator gaining full administrator privileges to the host.

2.2.3 Availability Impact (A)

This metric measures the potential impact to availability of a weakness. Availability refers to the accessibility of information resources. Attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of a host. The possible values for this metric are listed in Table 6. Increased availability impact increases the score.

Table 6. Availability Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the availability of the host.
Partial (P)	There is reduced performance or interruptions in resource availability. Examples are a network stack setting that is disabled, which if enabled would reduce the impact of denial of service attacks; and excess privileges that permit a user to stop services. Another example is the unavailability of a single necessary service, such as logging or auditing services.
Complete (C)	There is a total shutdown of the affected resource. The exploiter can render the resource completely unavailable. An example is excess privileges that permit a user to shut down a host or delete critical system files that the host cannot operate without. Another example is disabling a service that is needed to boot the host.

2.3 Base Vector

The CCSS vector facilitates the "open" nature of the framework. This vector contains the values assigned to each metric, and it is used to communicate exactly how the score for each configuration issue is derived. Therefore, the vector should always be presented with the score.

Each metric in the base vector consists of the abbreviated metric name, followed by a ":" (colon), then the abbreviated metric value. The vector lists these metrics in a predetermined order, using the "/" (slash) character to separate the metrics. The base vector structure is:

AV:[L,A,N]/AC:[H,M,L]/Au:[M,S,N]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C]

For example, a configuration issue with base metric values of "Access Vector: Low, Access Complexity: Medium, Authentication: None, Confidentiality Impact: None, Integrity Impact: Partial, Availability Impact: Complete" would have the following base vector: "AV:L/AC:M/Au:N/C:N/I:P/A:C".

2.4 Base Equation

The scoring equation and algorithms for the base metrics are described below. Further discussion of the origin and testing of the equation is available at http://www.first.org/cvss/.

The base equation is the foundation of CCSS scoring. The base equation is:

BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)–1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20* AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

The possible values for the base metrics are:

AccessVector = case AccessVector of

- requires local access: 0.395
- adjacent network accessible: 0.646

- network accessible: 1.0

AccessComplexity = case AccessComplexity of

- high: 0.35
- medium: 0.61
- low: 0.71

Authentication = case Authentication of

- requires multiple instances of authentication: 0.45
- requires single instance of authentication: 0.56
- requires no authentication: 0.704

ConfImpact = case ConfidentialityImpact of

- none: 0.0
- partial: 0.275
- complete: 0.660

IntegImpact = case IntegrityImpact of

- none: 0.0
- partial: 0.275
- complete: 0.660

AvailImpact = case AvailabilityImpact of

- none: 0.0
- partial: 0.275
- complete: 0.660

3. Scoring Guidelines

This section provides guidelines that should help analysts when scoring security configuration issues. The main difference between scoring security-related software flaws and configuration issues is that there is a single vector and base score for each software flaw, but potentially multiple vectors and base scores for a single configuration issue. Multiple vectors and base scores may be needed for a configuration issue if there is more than one way in which it can be configured with a negative security impact. For example, some Windows platforms have an Application Management service that affects the installation and use of applications. If the service is disabled, it prevents local users from installing and using new applications; if the service is enabled, it allows users to install or remove applications. Because these two cases have different security implications, an analyst would create a vector and a score for each case. In this example, and for many others, there is not necessarily a clear "correct" value, and each organization generally determines its own configuration requirements. An example is the length of time to lock out an account after too many failed login attempts. An analyst would generate two vectors and scores for this—one for the actual value being higher than policy and one for the actual value being lower than policy—and end users would select the appropriate vector and base score for each situation based on the organization's policy and actual host settings.

3.1 General

SCORING TIP #1: Configuration issue scoring should not take into account any interaction with other configuration issues or software flaws. That is, each configuration issue should be scored independently. However, if a configuration issue is necessarily dependent on another configuration setting, such as setting A is only used by a host if setting B is enabled, then analysts should assume that the other settings are configured so as to make the issue of interest relevant. For example, if Web server settings are only used by a host if the Web server service is enabled, then analysts should assume the service is enabled.

SCORING TIP #2: When scoring a configuration issue, consider the direct impact to the target host only. For example, consider a configuration issue that allows users to place files of their choice (such as malware) in a network share: the impact to the hosts of users that download and execute the malware could be much greater than the impact to the target host. However, this is an indirect impact, and should not be considered in scoring.[6]

SCORING TIP #3: Because a configuration issue does not state what the desired configuration is, analysts need to consider the plausible possibilities. For an issue with a small number of possible options, such as enabled/disabled, low/medium/high, or 1 through 5, analysts should consider the security implications of each option. For example, in the simplest situation—a setting that can either be enabled or disabled—an analyst should think about the security implications of two cases: 1) the setting is enabled but should be disabled, and 2) the setting is disabled but should be enabled. Only the cases that have a security impact should be analyzed further. Analysts should also eliminate cases that are considered extremely unlikely to occur. For each remaining case, analysts should create a vector and calculate a base score.

For an issue with a large number of possible settings, such as file privileges for users or the number of seconds for a timeout, it is not feasible to consider all the possible combinations of settings. For example, a host could be set to grant one set of excess privileges to user A, grant a different set of excess privileges

[6] While scoring indirect impact would be useful, it has proven difficult to calculate it accurately. In many cases, the indirect impact is dependent on the configurations and vulnerabilities of other arbitrary hosts. Since the potential impact on these systems is unknown, either the scoring would be unknown or some default scoring would have to be used, such as worst-case (i.e., assuming a complete impact). These scoring options are not helpful in accurately calculating impact, so indirect impact is omitted from base scoring.

to user B, and grant insufficient privileges to user C. However, we cannot score infinite possibilities for an issue, so we instead consider the common cases independently. For example, if the issue involved privileges for a system binary used for system management, the case most likely to occur is that users can execute the binary but should not be able to (since it is intended for system administrators). Other broad cases with interesting security implications are users being able to modify or delete the system binary, and no users (including administrators) being able to run the binary. Another example is for a timeout; analysts should consider two cases—the timeout being set too high and too low. After analyzing the selected cases and combinations of cases, the analyst should create a vector for each and calculate a score.

An organization that wishes to use CCSS scoring for the configuration issues within its own hosts would compare the requirements specified in its own policies and security configuration checklists against the hosts' actual security settings to identify configuration discrepancies and then select the appropriate CCSS vector and score for each configuration issue.

SCORING TIP #4: Many applications, such as Web browsers, can be run with different privileges, and scoring the impact involves making an assumption as to what privileges are used. Therefore, in cases where assumptions about application privileges must be made to determine a score, configuration issues should be scored according to the privileges most commonly used. This may not necessarily reflect security best practices, especially for client applications which are often run with root-level privileges. When uncertain as to which privileges are most common, scoring analysts should assume a default configuration.

3.2 Base Metrics

SCORING TIP #5: When a configuration issue can be exploited both locally and from the network, the "Network" value should be chosen. When a configuration issue can be exploited both locally and from adjacent networks, but not from remote networks, the "Adjacent Network" value should be chosen. When a configuration issue can be exploited from the adjacent network and remote networks, the Access Vector metric should be assigned a value of "Network".

SCORING TIP #6: Some software, particularly client applications and utilities, may have local configuration issues that can be exploited remotely either through user-complicit actions or via automated processing. For example, decompression utilities and virus scanners automatically scan incoming email messages. If a configuration issue caused these to ignore certain types of content, then that issue should have its Access Vector metric assigned a value of "Network".

SCORING TIP #7: If the configuration issue exists in an authentication scheme itself (e.g., PAM, Kerberos) or an anonymous service (e.g., public FTP server), the Authentication metric should be scored as "None" because the exploiter can exploit the issue without supplying valid credentials. Presence of a default user account may be considered as "Single" or "Multiple" Authentication (as appropriate), but would have Access Complexity of "Low" if the credentials are publicized (which is usually the case).

SCORING TIP #8: Configuration issues that give root-level access should be scored with complete loss of confidentiality, integrity, and availability, while issues that give user-level access should be scored with only partial loss of confidentiality, integrity, and availability. For example, an issue that allows an attacker to modify an operating system password file as desired (which would permit the attacker to change the root password or create a new root-level account) should be scored with complete impact of confidentiality, integrity, and availability. On the other hand, an issue that enables an attacker to impersonate a valid user who has limited privileges should be scored with a partial impact of confidentiality, integrity, and availability.

SCORING TIP #9: Configuration issues that can permit a partial or complete loss of integrity often also permit an impact to availability. For example, an exploiter who is able to modify records can probably also delete them.

SCORING TIP #10: Configuration issues that make it more difficult to detect security violations, such as issues that cause certain types of security activities not to be logged, should be scored at a minimum with partial loss of integrity.

4. Scoring Examples

This section provides examples of how CCSS would be used for several types of configuration issues.[7] The issues in the examples are from the Common Configuration Enumeration (CCE) version 5 standard[8], which assigns unique identifiers to security configuration issues for operating systems and applications.

4.1 CCE-4675-5

Consider CCE-4675-5 for Sun Solaris 10: "Kernel level auditing should be enabled or disabled as appropriate." If auditing should be enabled but is not, a wide range of events will not be logged, including login/logout events, administrative actions, file attribute modifications, and process events.

Since some of the types of events logged in kernel level auditing are remotely triggered, the Access Vector is "Network". The Access Complexity is "Low" because no action is needed; the events automatically fail to be logged for all users. No authentication is required to trigger the weakness, so the Authentication metric is "None". The failure to log kernel level events is a partial compromise of system integrity. There is no impact on confidentiality or availability.

The base vector for this weakness is AV:N/AC:L/Au:N/C:N/I:P/A:N. This vector produces a base score of 5.0.

4.2 CCE-4693-8

Consider CCE-4693-8 for Sun Solaris 10: "File permissions for the /etc/cron.d/cron.allow file should be configured correctly." This file lists usernames that are permitted to use cron, which allows users to have commands automatically run at a certain time.

Access to this file requires a local account on the computer, so the Access Vector is "Local". The Access Complexity varies by case (see below). No additional authentication other than the initial target access is required to trigger the weakness, so the Authentication metric is "None".

Categorizing the impact is challenging because there are so many possibilities for the permissions for the file. We do not know who has access or should have access, or what types of access are needed or have been granted. However, the primary threat seems to be modifying the file we can assume that not giving administrators sufficient privileges to access the file is not a security concern because administrators could grant themselves access to the file as needed. So in terms of security threats, we can consider two cases: 1) a user should be able to modify cron.allow but cannot, and 2) a user should not be able to modify cron.allow but can.

For case 1, the impact would be rated as a partial compromise of availability, because the functionality is not available, and no compromise of confidentiality or integrity. The Access Complexity is "Low" because the user simply attempts to access the cron.allow file. The base vector for case 1 is AV:L/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 2.1.

For case 2, the impact would be rated as a partial compromise of integrity because the user could alter the contents of cron.allow. The potential future impact of this change, such as permitting users to use cron without authorization or preventing authorized users from using cron, is not considered because it is an indirect impact; only the direct impact is analyzed for the score. The Access Complexity is "Low"

[7] The scores were calculated using the National Vulnerability Database CVSS 2.0 calculator available online at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2.

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

because the user simply accesses the cron.allow file. The base vector for case 2 is AV:L/AC:L/Au:N/C:P/I:P/A:P. This vector produces a base score of 4.6.

4.3 CCE-2786-2

Consider CCE-2786-2 for Windows XP: “The ‘create a pagefile’ user right should be assigned to the correct accounts.” This weakness gives users the ability to create pagefiles and to alter their sizes, including setting their size to 0. (We can assume that not giving authorized users the right to create a pagefile is of trivial security significance, since pagefiles need to be created so rarely, so we will not analyze that case.) We do not know to whom the access has been granted.

Access to this file requires a local account on the computer, so the Access Vector is "Local". The Access Complexity is "Low" because users can simply use regular OS features to create the pagefile. No additional authentication other than the initial target access is required to use the weakness, so the Authentication metric is "None".

Although we do not know which parties have gained this access, for the purpose of rating impact it is largely irrelevant. Giving this privilege to any users is a partial compromise of host availability, because having small pagefiles or no pagefile could seriously impact host performance. There is no impact on confidentiality or integrity.

The base vector for this weakness is: AV:L/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 2.1.

4.4 CCE-2363-0

Consider CCE-2363-0 for Windows Vista: “The ‘account lockout duration’ policy should meet minimum requirements.” By having a shorter-than-recommended lockout duration time, this weakness could allow attackers to guess passwords more frequently. By having a longer-than-recommended lockout duration time, this weakness could delay users who have been locked out from regaining access to their accounts; on Windows hosts, if it is set to 0, the account will remain locked until an administrator unlocks it. Since both cases have security implications and the value assigned to the policy is subjective, both will be analyzed.

For the first case (shorter-than-recommended lockout), the Access Vector is “Local”. The Access Complexity is “High” because the weakness only makes it slightly easier for an attacker to guess passwords. No authentication is needed, so the Authentication metric is “None”. Because the weakness makes it easier for attackers to gain access to passwords, we rate the impact as a partial compromise of host confidentiality. There is no impact on integrity or availability because further impact on the host would be secondary effects that occur once the attacker is actually able to log onto it. The base vector is: AV:L/AC:H/Au:N/C:P/I:N/A:N. This vector produces a base score of 1.2.

For the second case (longer-than-recommended lockout), the Access Vector is “Local”. The Access Complexity is “Low” because accounts can easily be locked out, either accidentally or intentionally. No authentication is needed, so the Authentication metric is “None”. Because the weakness can cause users to be prevented from logging in, we rate the impact as a partial compromise of host availability. There is no impact on confidentiality or integrity. The base vector is: AV:L/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 2.1.

4.5 CCE-2366-3

Consider CCE-2366-3 for Windows XP: "The 'shut down the system' user right should be assigned to the correct accounts." We do not know to whom the access has been granted.

The obvious case is that users have permission to shut down the host, but should not. Access to this file requires a local account on the computer, so the Access Vector is "Local". The Access Complexity is "Low" because the user can simply use regular OS features to shut down the host. No additional authentication other than the initial target access is required to use the weakness, so the Authentication metric is "None". Giving this privilege to any users is a full compromise of host availability (the host can be shut down by users at will). There is no impact on confidentiality or integrity. The base vector for this weakness is: AV:L/AC:L/Au:N/C:N/I:N/A:C. This vector produces a base score of 4.9.

Another case with possible security implications is if no users, including administrators, had the right to shut the host down. This could delay administrators in shutting down the host when needed, so it would be a partial compromise of host availability. The base vector would be AV:L/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 2.1.

4.6 CCE-4208-5

CCE-4208-5 for Internet Explorer 7 is: "The 'Disable Offline Page Hit Logging' setting should be configured correctly." The possible options for this setting are Enabled and Disabled. This setting affects whether or not a remote Web site can get data from the local host about the local host's offline access to the remote Web site. If the setting is enabled, then the Web site cannot get the usage data; if the setting is disabled, then the Web site can get the usage data. From the perspective of the local host, the security issue would be unwanted exposure of the usage data to the remote Web site. The case where the Web site cannot get the usage data is, from the perspective of the local host, an operational issue, not a security issue.

For this case, the Access Vector is "Network" because remote Web sites can access the usage data. However, the Access Complexity is "High" because the weakness can only be used by Web sites that the user has configured to have stored offline and then actually accessed. No authentication is required, so the Authentication metric is "None". The exposure of the user's data is a partial compromise of confidentiality, and has no impact on integrity or availability.

The base vector would be AV:N/AC:H/Au:N/C:P/I:N/A:N. This vector produces a base score of 2.6.

4.7 CCE-2519-7

CCE-2519-7 for Windows Vista is: "The amount of idle time required before disconnecting a session should be set correctly." This can be set to a number of minutes.

If the setting is too low, sessions are disconnected more quickly than desired, which would cause a slight impact to availability. The Access Vector would be "Network" since these are remote sessions to the target. The Access Complexity would be "Low" since no special action is needed. Authentication would be "None". The base vector would be AV:N/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 5.0.

If the setting is too high, sessions are disconnected more slowly than desired, which could possibly prevent another user from initiating a session if the host can only support a certain number of connections and they are all taken (not being disconnected promptly when idle). This is also a slight impact to

availability. Having the setting too high is also a security issue because it could increase the likelihood of someone gaining unauthorized access to another user's idle session. That would give someone user-level access to the target, so the impact would be partial compromise of confidentiality, integrity, and availability. The Access Vector would have to be "Network" since this is a remote session to the target. The Access Complexity would be "High", because the attacker would first have to gain access to the remote host, and then misuse the remote session. Authentication should be "None" because no authentication to the target is needed from an established session. The base vector would be AV:N/AC:H/Au:N/C:P/I:P/A:P. This vector produces a base score of 5.1.

4.8 CCE-3171-6

CCE-3171-6 for Windows XP is: "Application Layer Gateway Service". This service can be enabled or disabled. The major security implication of this service is that if it is disabled, the built-in firewall will not start, which could permit remote hosts to gain unauthorized access to network services on the target that are otherwise blocked by the firewall. This is a partial compromise of availability (because it prevents the firewall from being used) and confidentiality (because an attacker can gain unauthorized access to services); it does not impact host integrity. On Windows hosts, having the application layer gateway service disabled also prevents the Internet Connection Sharing feature from being used, which prevents the target from providing Internet access to other local hosts, causing a partial impact on availability.

Because remote hosts could gain unauthorized access, the Access Vector is "Network". The Access Complexity would be "Low" because the attacker would simply have to initiate a standard connection to the target. Authentication should be "None" because we are assuming that it is likely that at least some of the exposed network services do not require authentication or reveal information before requiring authentication (e.g., version information in logon banners).

The base vector would be AV:N/AC:L/Au:N/C:P/I:N/A:P. This vector produces a base score of 6.4.

4.9 CCE-3047-8

CCE-3047-8 for Windows XP is: "Application Management". This service can be enabled or disabled. There are security implications to both cases. If this service is disabled but should be enabled, it prevents local users from installing and using new applications, which has a partial impact on availability. If this service is enabled but should be disabled, it allows a user to install or remove programs, which could alter the host's integrity. Both cases have an Access Vector of "Local", Access Complexity of "Low", and Authentication of "None".

The base vector for the first case is AV:L/AC:L/Au:N/C:N/I:N/A:P, with a base score of 2.1. The base vector for the second case is AV:L/AC:L/Au:N/C:N/I:P/A:N, also with a base score of 2.1.

4.10 CCE-4191-3

CCE-4191-3 in Red Hat Enterprise Linux 5 is: "The dhcp client service should be enabled or disabled as appropriate for each interface". There are security implications to both enabling and disabling the service. If the DHCP client is disabled but should be enabled, then the host may not be able to get an IP address, thus preventing use of IP services—partial impact on availability. If the DHCP client is enabled but should be disabled, then the host may be able to get access to IP services that it should not be able to—a partial impact on confidentiality—although arguably a host could just set its own IP address and get access to IP services with or without a DHCP client. The second case would be rare compared to the first case, and arguably the second case should not even be considered.

In both cases, the Access Vector would be “Local” (the DHCP client initiates all requests). The Access Complexity is “Low” and Authentication is set to “None”.

The base vector for the first case (DHCP disabled but should be enabled) is AV:L/AC:L/Au:N/C:N/I:N/A:P, with a base score of 2.1. The base vector for the second case (DHCP enabled but should be disabled) is AV:L/AC:L/Au:N/C:P/I:N/A:N, also with a base score of 2.1.

4.11 CCE-3245-8

CCE-3245-8 in Windows XP is: “IPSEC Services”. This service can be enabled or disabled. This service can protect the confidentiality and integrity of network traffic, and some IPsec services (such as on Windows hosts) also perform packet filtering (firewall capabilities). So if the service is disabled but should be enabled, it could expose the host’s network services to unauthorized access from other hosts (partial impact on confidentiality) and also permit network traffic to be transmitted without expected confidentiality or integrity protections (partial impact on confidentiality and integrity). Some network communications may require IPsec protection, so in those cases the traffic could not be sent (partial impact on availability).

The Access Vector would be “Network” because other hosts could potentially access network services, other hosts’ network traffic might go unprotected, and other hosts might be unable to communicate with the host (e.g., if IPsec is required for communications but unavailable). Access Complexity is “Low” because no specific actions need to be performed, and Authentication is “None”.

The base vector would be AV:N/AC:L/Au:N/C:P/I:P/A:P. This vector produces a base score of 7.5.

5. Appendix A—Additional Resources

The following are resources related to CCSS and the CVSS specification from which it is derived.

- More information on CCE is available at http://cce.mitre.org/.
- CVSS calculators can be used to calculate base CCSS scores since they use the same metric values and formula. The NIST CVSS calculator can be found at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2.
- The CVSS version 2 specification is available at http://www.first.org/cvss/cvss-guide.html. General information on CVSS's development is documented at http://www.first.org/cvss/.
- NISTIR 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*, describes the CVSS version 2 specification and also provides insights as to how CVSS scores can be customized for Federal agency-specific purposes. The report is available at http://csrc.nist.gov/publications/PubsNISTIRs.html.

6. Appendix B—Acronyms and Abbreviations

This appendix contains selected acronyms and abbreviations used in the publication.

A Adjacent Network
A Availability
AC Access Complexity
AU Authentication
AV Access Vector

C Complete or Confidentiality
CCE Common Configuration Enumeration
CCSS Common Configuration Scoring System
CVSS Common Vulnerability Scoring System

DHCP Dynamic Host Configuration Protocol

FIRST Forum of Incident Response and Security Teams
FTP File Transfer Protocol

H High

I Integrity
IEEE Institute of Electrical and Electronics Engineers
IP Internet Protocol
IPsec Internet Protocol Security
IR Interagency Report
IT Information Technology
ITL Information Technology Laboratory

L Local or Low
LAN Local Area Network

M Medium or Multiple

N Network or None
NIST National Institute of Standards and Technology
NISTIR National Institute of Standards and Technology Interagency Report
NVD National Vulnerability Database

OS Operating System

P Partial
PAM Pluggable Authentication Module

S Single
SMTP Simple Mail Transfer Protocol

USB Universal Serial Bus