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

NIST Interagency Report 7502
(Second Public Draft)

The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities (DRAFT)

Karen Scarfone
Peter Mell

NIST Interagency Report 7502
(Second Public Draft)

The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities (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

June 2009

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, 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)
42 pages (Jun. 2009)

Acknowledgments

The authors, Karen Scarfone and Peter Mell of the National Institute of Standards and Technology (NIST), sincerely appreciate the contributions of the Forum of Incident Response and Security Teams (FIRST) Common Vulnerability Scoring System (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. The authors are also grateful for the contributions of Elizabeth Van Ruitenbeek of the University of Illinois at Urbana-Champaign in shaping and refining the NIST approach to vulnerability measurement and scoring.

Portions of this report are based on the official Common Vulnerability Scoring System (CVSS) standard[1] from the CVSS Special Interest Group; on NIST Interagency Report (IR) 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*[2]; and on draft NIST IR 7517, *The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities*[3].

Abstract

The Common Configuration Scoring System (CCSS) is a set of measures of the severity of software security configuration issues. CCSS is derived from CVSS, which was developed to measure the severity of vulnerabilities due to software flaws. CCSS can assist organizations in making sound decisions as to how security configuration issues should be addressed and can provide data to be used in quantitative assessments of the overall security posture of a system. This report defines proposed measures for CCSS and equations to be used to combine the measures into severity 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 security configuration issues.

Audience

This report is directed primarily at information security researchers, particularly those interested in vulnerability measurement or security automation; security product vendors; and vulnerability analysts. The report is also intended to make organizations aware of the existence of CCSS and illustrate how they could use CCSS once repositories of measures and scores are publicly available.

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

Table of Contents

List of Tables

1. Overview of Vulnerability Measurement and Scoring

This section provides an overview of vulnerability measurement and scoring. It first defines the major categories of system vulnerabilities. Next, it discusses the need to measure the characteristics of vulnerabilities and generate scores based on those measurements. Finally, it introduces recent vulnerability and measurement scoring systems.

1.1 Categories of System Vulnerabilities

There are many ways in which the vulnerabilities of a system can be categorized. For the purposes of vulnerability scoring, this report uses three high-level vulnerability categories: software flaws, security configuration issues, and software feature misuse.[4] These categories are described below.

A *software flaw vulnerability* is caused by an unintended error in the design or coding of software. An example is an input validation error, such as user-provided input not being properly evaluated for malicious character strings and overly long values associated with known attacks. Another example is a race condition error that allows the attacker to perform a specific action with elevated privileges.

A security configuration setting is an element of a software's security that can be altered through the software itself. Examples of settings are an operating system offering access control lists that set the privileges that users have for files, and an application offering a setting to enable or disable the encryption of sensitive data stored by the application. A *security configuration issue vulnerability* involves the use of security configuration settings that negatively affect the security of the software.

A software feature is a functional capability provided by software. A *software feature misuse vulnerability* is a vulnerability in which the feature also provides an avenue to compromise the security of a system. These vulnerabilities are caused by the software designer making trust assumptions that permit the software to provide beneficial features, while also introducing the possibility of someone violating the trust assumptions to compromise security. For example, email client software may contain a feature that renders HTML content in email messages. An attacker could craft a fraudulent email message that contains hyperlinks that, when rendered in HTML, appear to the recipient to be benign, but actually take the recipient to a malicious web site when they are clicked on. One of the trust assumptions in the design of the HTML content rendering feature was that users would not receive malicious hyperlinks and click on them.

Software feature misuse vulnerabilities are introduced during the design of the software or a component of the software (e.g., a protocol that the software implements). Trust assumptions may have been explicit—for example, a designer being aware of a security weakness and determining that a separate security control would compensate for it. However, trust assumptions are often implicit, such as creating a feature without first evaluating the risks it would introduce. Threats may also change over the lifetime of software or a protocol used in software. For example, the Address Resolution Protocol (ARP) trusts that an ARP reply contains the correct mapping between Media Access Control (MAC) and Internet Protocol (IP) addresses. The ARP cache uses that information to provide a useful service—to enable sending data between devices within a local network. However, an attacker could generate false ARP messages to poison a system's ARP table and thereby launch a denial-of-service or a man-in-the-middle attack. The ARP protocol was standardized over 25 years ago[5], and threats have changed a great deal since then, so the trust assumptions inherent in its design then are unlikely to still be reasonable today.

---

[4] There are other types of vulnerabilities, such as physical vulnerabilities, that are not included in these categories.

[5] David Plummer, Request for Comments (RFC) 826, *An Ethernet Resolution Protocol* (http://www.ietf.org/rfc/rfc826.txt)

It may be hard to differentiate software feature misuse vulnerabilities from the other two categories. For example, both software flaws and misuse vulnerabilities may be caused by deficiencies in software design processes. However, software flaws are purely negative—they provide no positive benefit to security or functionality—while software feature misuse vulnerabilities occur as a result of providing additional features.

There may also be confusion regarding misuse vulnerabilities for features that can be enabled or disabled—in a way, configured—versus security configuration issues. The key difference is that for a misuse vulnerability, the configuration setting enables or disables the entire feature, and does not specifically alter just its security; for a security configuration issue vulnerability, the configuration setting alters only the software's security. For example, a setting that disables all use of HTML in emails has a significant impact on both security and functionality, so a vulnerability related to this setting would be a misuse vulnerability. A setting that disables the use of an antiphishing feature in an email client has a significant impact on only security, so a vulnerability with that setting would be considered a security configuration issue vulnerability.

1.2 The Need for Vulnerability Measurement and Scoring

No system is 100% secure: every system has vulnerabilities. At any given time, a system may not have any known software flaws, but security configuration issues and software feature misuse vulnerabilities are always present. Misuse vulnerabilities are inherent in software features because each feature must be based on trust assumptions—and those assumptions can be broken, albeit involving significant cost and effort in some cases. Security configuration issues are also unavoidable for two reasons. First, many configuration settings increase security at the expense of reducing functionality, so using the most secure settings could make the software useless or unusable. Second, many security settings have both positive and negative consequences for security. An example is the number of consecutive failed authentication attempts to permit before locking out a user account. Setting this to 1 would be the most secure setting against password guessing attacks, but it would also cause legitimate users to be locked out after mistyping a password once, and it would also permit attackers to perform denial-of-service attacks against users more easily by generating a single failed login attempt for each user account.

Because of the number of vulnerabilities inherent in security configuration settings and software feature misuse possibilities, plus the number of software flaw vulnerabilities on a system at any given time, there may be dozens or hundreds of vulnerabilities on a single system. These vulnerabilities are likely to have a wide variety of characteristics. Some will be very easy to exploit, while others will only be exploitable under a combination of highly unlikely conditions. One vulnerability might provide root-level access to a system, while another vulnerability might only permit read access to an insignificant file. Ultimately, organizations need to know how difficult it is for someone to exploit each vulnerability and, if a vulnerability is exploited, what the possible impact would be.

If vulnerability characteristics related to these two concepts were measured and documented in a consistent, methodical way, the measurements could be analyzed to determine which vulnerabilities are most important for an organization to address using its limited resources. For example, an organization could measure the relative severity of software flaws to help determine which should be patched as quickly as possible and which can wait until the next regularly scheduled outage window. When planning the security configuration settings for a new system, an organization could use vulnerability measurements as part of determining the relative importance of particular settings and identifying the settings causing the greatest increase in risk. Vulnerability measurement is also useful when evaluating the security of software features, such as identifying the vulnerabilities in those features that should have compensating controls applied to reduce their risk (for example, antivirus software to scan email

attachments and awareness training to alter user behavior) and determining which features should be disabled because their risk outweighs the benefit that they provide.

There are additional benefits to having consistent measures for all types of system vulnerabilities. Organizations can compare the relative severity of different vulnerabilities from different software packages and on different systems. Software vendors can track the characteristics of a product's vulnerabilities over time to determine if its security is improving or declining. Software vendors can also use the measures to communicate to their customers the severity of the vulnerabilities in their products. Auditors and others performing security assessments can check systems to ensure that they do not have unmitigated vulnerabilities with certain characteristics, such as high impact measures or high overall severity scores.

Although having a set of measures for a vulnerability provides the level of detail necessary for in-depth analysis, sometimes it is more convenient for people to have a single measure for each vulnerability. So quantitative measures can be combined into a score—a single number that provides an estimate of the overall severity of a vulnerability. Vulnerability scores are not as quantitative as the measures that they are based on, so they are most helpful for general comparisons, such as a vulnerability with a score of 10 (on a 0 to 10 scale) being considerably more severe than a vulnerability with a score of 2. Small scoring differences, such as vulnerabilities with scores of 4.8 and 5.1, do not necessarily indicate a significant difference in severity because of the margin of error in individual measures and the equations that combine those measures.

1.3 Vulnerability Measurement and Scoring Systems

To provide standardized methods for vulnerability measurement and scoring, three specifications have been created, one for each of the categories of system vulnerabilities defined in Section 1.1. The first specification, the Common Vulnerability Scoring System (CVSS), addresses software flaw vulnerabilities. The first version of CVSS was introduced in 2004, and the second version became available in 2007.[6] CVSS has been widely adopted by the Federal government, industry, and others. CVSS was originally intended for use in prioritizing the deployment of patches, but there has been considerable interest in applying it more broadly, such as using its measures as inputs to risk assessment methodologies.

The second vulnerability measurement and scoring specification is the Common Misuse Scoring System (CMSS). CMSS was designed for measuring and scoring misuse vulnerabilities. CMSS uses the basic components of CVSS and adjusts them to account for the differences between software flaws and misuse vulnerabilities. A draft of the CMSS specification was released for public comment in February 2009 and is undergoing revisions as of this writing.[7]

The Common Configuration Scoring System (CCSS), the third of the vulnerability measurement and scoring specifications, is defined in this report. CCSS addresses software security configuration issue vulnerabilities. The original draft specification for CCSS, which was released for public comment in May 2008, was based solely on CVSS. After CMSS was developed, the CCSS specification was redesigned based on new concepts introduced in CMSS. So the current CCSS specification is largely based on CVSS and CMSS, and it is intended to complement them.

---

[6] The official CVSS version 2 specification is available at http://www.first.org/cvss/cvss-guide.html. NIST has also published a Federal agency-specific version of the specification in NIST IR 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems* (http://csrc.nist.gov/publications/PubsNISTIRs.html).

[7] Draft NIST IR 7517, *The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities*

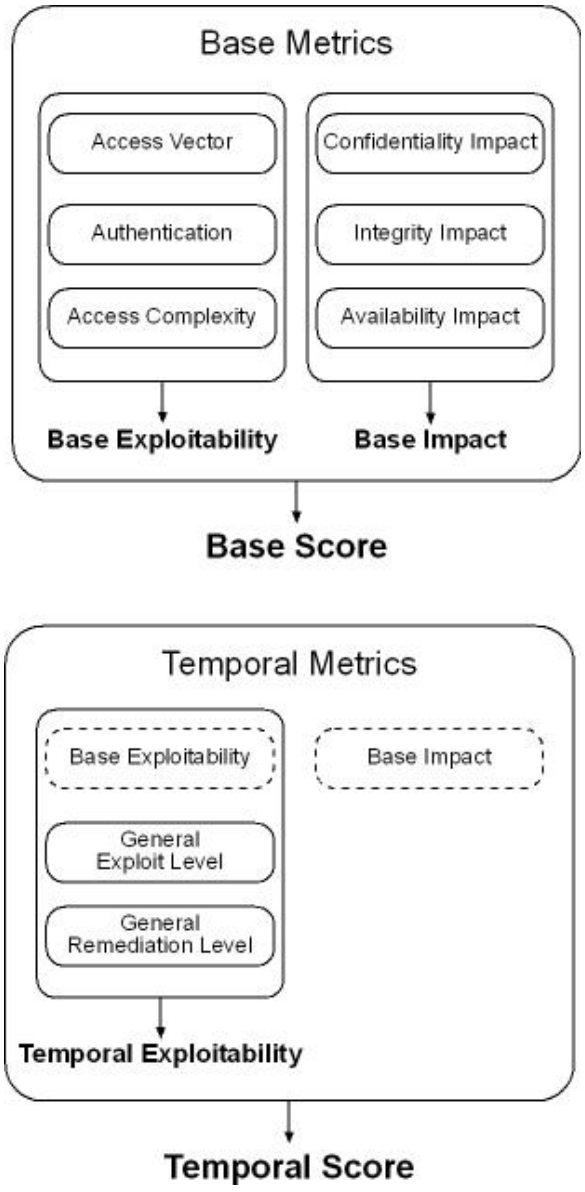
The three vulnerability measurement and scoring systems are similar. They all use the same six core measures to capture the fundamental characteristics of vulnerabilities. They all generate vulnerability severity scores in the range of 0 (lowest severity) to 10 (highest severity). However, there are also some significant differences in the three specifications. These differences are discussed in Section 4, after the CCSS specification has been defined and illustrated in Sections 2 and 3.

CCSS can help organizations to better analyze the risk inherent in their security configurations. Each security configuration decision an organization makes can have positive and negative effects of varying degrees to the security of a system. 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 system. 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 could be exploited 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.

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 system 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 Metrics

This section defines the metrics that comprise the proposed CCSS specification. The CCSS metrics are organized into three groups: base, temporal, and environmental. Base metrics describe the characteristics of a configuration issue that are constant over time and across user environments. Temporal metrics describe the characteristics of configuration issues that can change over time but remain constant across user environments. Environmental metrics are used to customize the base and temporal scores based on the characteristics of a specific user environment. Figure 1 shows how the base, temporal, and environmental scores are calculated from the three groups of metrics.



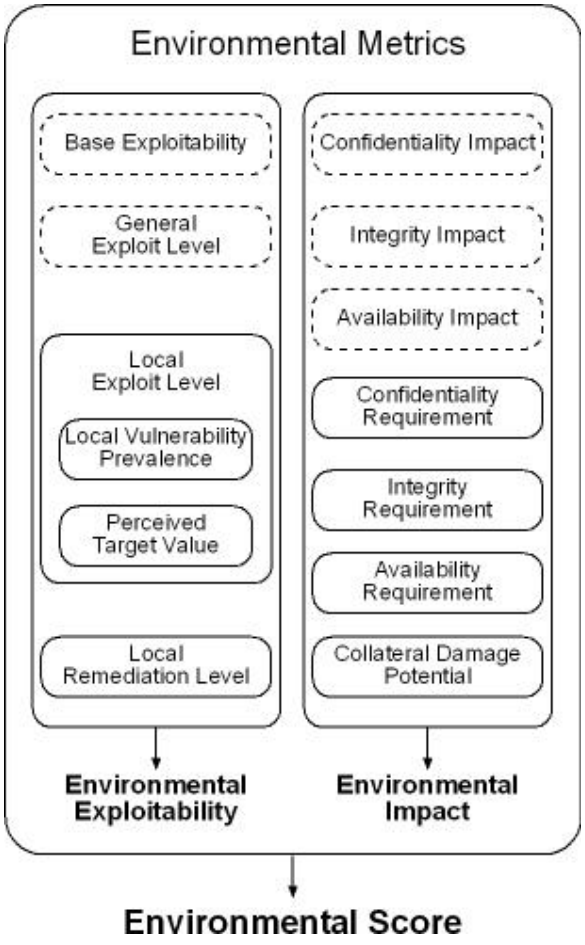


Figure 1. CCSS Metric Groups

2.1 Base Metrics

This section describes the base metrics, which measure the characteristics of a security configuration issue that are constant with time and across user environments. The base metrics measure two aspects of vulnerability severity: Exploitability and Impact.

2.1.1 Exploitability

The Exploitability of a security configuration issue can be captured using the Access Vector, Authentication, Access Complexity, and Exploitation Method metrics. These metrics are adapted from the CVSS and CMSS specifications and reinterpreted in the context of security configuration issues.

2.1.1.1 Exploitation Method (EM)

Weaknesses caused by security configuration issues can be taken advantage of in two ways: actively and passively.[8] These are indicated in CCSS as an Exploitation Method metric of Active (A) or Passive (P). Some weaknesses can be actively exploited, such as an attacker gaining access to a sensitive file because the targeted system is incorrectly configured to permit any user to read the file. Other weaknesses

[8] 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 to date.

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.

The Exploitation Method metric is not used directly in generating CCSS scores, but the definitions for some of the Exploitability base metrics are dependent on the value of the Exploitation Method metric. The Exploitation Method metric is also useful for planning mitigation strategies, because most Passive configuration issues involve an error or policy violation that can be fixed quickly. Active configuration issues have a much wider range of mitigation possibilities and often require significantly more analysis and planning for mitigation than Passive configuration issues.

Some organizations also find the Exploitation Method metric valuable because of how they define availability. For example, an organization might consider a security configuration error that prevents use of a particular system function as an operational concern, not a security concern, and prioritize and remediate it differently. Such errors would all have an Exploitation Method metric value of Passive, so organizations could use the metric to quickly differentiate the configuration issues that they consider operational from the others.

2.1.1.2 Access Vector (AV)

The Access Vector metric reflects the access required to exploit the configuration issue. To produce an Access Vector score for a configuration issue, consider what access to the system the exploiter[9] must possess in order to exploit the issue. The possible values for this metric are listed in Table 1. The more remote an exploiter can be, the greater the vulnerability score.

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 may be remote from the affected system.

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

[9] 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 vulnerable system 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 vulnerable software. Examples of 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 systems on the broadcast or collision domain of the software. An example of such a configuration issue is a system that is intended to share its Internet access with other systems 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 that 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 systems outside the broadcast or collision domain. An example is a system that is intended to provide a network service to all other systems, but the service has inadvertently been disabled. Another example is a system that interacts with remote systems, but has all of its logging and auditing capabilities disabled; the system will fail to record events involving the remote systems, so this could permit malicious activity by those remote systems to go unnoticed.

2.1.1.3 Authentication (Au)

The Authentication 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 2. 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 system 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 system. 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[10] 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 2. 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 hosted on that system.
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, suppose that a mail server is configured to permit use of an optional command that reveals server configuration information. If this command can be used by a person without first authenticating to the mail server, the metric should be set to "None" because the exploiter can launch the exploit before credentials are required. If the command can only be used after successful authentication to the mail server, then the weakness should be set to "Single" or "Multiple," depending on how many instances of authentication must occur before issuing the command.

2.1.1.4 Access Complexity (AC)

The Access Complexity metric reflects the complexity of the attack required to exploit the configuration issue. 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 system. For example, consider a default user account and password for a network service: once the target system 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 3. The lower the required complexity, the higher the vulnerability score.

[10] For the purposes of this report, a user is a person or entity whose direct actions take advantage of the software security configuration issue. A user may be malicious or non-malicious. In contrast, an attacker is always malicious.

Table 3. 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 exploiter 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 systems 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 systems 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.

2.1.2 Impact

The Impact of a software security configuration issue can be captured using the Confidentiality Impact, Integrity Impact, and Availability Impact metrics. These metrics are adapted from the CVSS specification and reinterpreted in the context of security configuration issues. These three Impact metrics measure how exploitation of a configuration issue could directly affect a targeted system. The Impact metrics independently reflect the degree of loss of confidentiality, integrity, and availability. For example, exploitation of a particular configuration issue could cause a partial loss of integrity and availability, but no loss of confidentiality.

For many configuration issues, the possible impact is dependent on the privileges held by the user or application (including services). For example, a user account could have restricted privileges or full root-level privileges; exploiting many vulnerabilities would cause a greater impact if full privileges were available. However, the privileges available through exploitation are environment-specific. CCSS

prevents environment-specific privileges from affecting the base impact metrics by assuming for those metrics that the generally accepted best practices for the privileges are being employed.[11]

CCSS allows organizations to take into account different privileges that they may provide in their environments. This is done through a combination of base and environmental metrics. The base metric, Privilege Level (PL), indicates the level of unauthorized access that an attacker could gain, such as impersonating a user or gaining full access to an application or an operating system. Possible values for Privilege Level are Root Level (R), User Level (U), Application Level (A), and Not Defined (ND). The environmental metrics that use the Privilege Level are described in Section 2.3.3.1.

2.1.2.1 Confidentiality Impact (C)

The Confidentiality Impact metric measures the potential impact on confidentiality of a successfully exploited security configuration issue. Confidentiality refers to limiting information access and disclosure and system access to only authorized users, as well as preventing access by, or disclosure to, unauthorized parties. The possible values for this metric are listed in Table 4. The greater the potential impact, the higher the score.

Table 4. Confidentiality Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the confidentiality of the system.
Partial (P)	There is considerable information disclosure. Access to some system files is possible, but the exploiter does not have control over what is obtained, or the scope of the loss is constrained. An example is a vulnerability that divulges only certain tables in a database. 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 unauthorized access to the system. Examples include an authorized user gaining access to certain prohibited system functions, an unauthorized user gaining access to a network service offered by the system, and an unauthorized user gaining user or application-level privileges on the system (such as a database administration account).
Complete (C)	There is total information disclosure, resulting in all system files being revealed. The exploiter is able to read all of the system'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 system.

2.1.2.2 Integrity Impact (I)

The Integrity Impact metric measures the potential impact to integrity of a successfully exploited security configuration issue. Integrity refers to the trustworthiness and guaranteed veracity of information. The possible values for this metric are listed in Table 5. The greater the potential impact, the higher the score.

[11] In cases where it is unclear what the best practice is for privileges, a default configuration is assumed.

Table 5. Integrity Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the integrity of the system.
Partial (P)	Modification of some system 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 system's security configuration can be altered. An example is a configuration setting that would allow an unauthorized file (such as one containing malware) to be stored on the system or allow an unauthorized program to be installed on the system.
Complete (C)	There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised. The exploiter is able to modify any files on the target system. An example is someone who is not authorized to act as a system administrator gaining full administrator privileges to the system.

2.1.2.3 Availability Impact (A)

The Availability Impact metric measures the potential impact to availability of a successfully exploited security configuration issue. Availability refers to the accessibility of information resources. Attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of a system. The possible values for this metric are listed in Table 6. The greater the potential impact, the higher the score.

Table 6. Availability Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the availability of the system.
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 system or delete critical system files that the system cannot operate without. Another example is disabling a service that is needed to boot the system.

2.2 Temporal Metrics

The threat posed by a security configuration issue may change over time. The base metrics are limited to the characteristics of a security configuration issue that are constant over time and across user environments. To incorporate the time-variant aspect of threats, the temporal metrics produce a scaling factor that is applied to the Exploitability components of the base metric. Temporal metrics describe the characteristics of security configuration issue vulnerabilities that can change over time but remain constant across user environments.

The two components of CMSS temporal metrics are the General Exploit Level and the General Remediation Level. Since temporal metrics are optional, each includes a default metric value that has no

effect on the score. This value is used when the scoring analyst wishes to ignore a particular metric because it does not apply or the analyst does not have sufficient data to determine the appropriate metric value.

2.2.1 General Exploit Level (GEL)

The General Exploit Level metric measures the prevalence of attacks against a security configuration issue—how often any vulnerable system is likely to come under attack. If a security configuration issue could be exploited more widely with the use of exploit code, the prevalence of attacks may be related to the current state of exploit techniques or exploit code availability. Public availability of easy-to-use exploit code increases the number of potential attackers by including those who are unskilled, thereby increasing the severity of the vulnerability. The availability of automated exploit code also increases the number of attacks each attacker can launch. However, note that attacks may not require exploit code. For example, consider a security configuration issue that can be attacked by sending a user an email with instructions to perform actions that result in an exploit. The prevalence of this type of attack would be measured by the frequency with which the exploit email is received by users on a typical vulnerable system.

The possible values for this metric are listed in Table 7. The more frequently exploitation attempts occur for a vulnerability, the higher the score.

Table 7. General Exploit Level Scoring Evaluation

Metric Value	Description
None (N)	Exploits have not yet been observed.
Low (L)	Exploits are rarely observed. Expected time-between-exploits for a vulnerable system is measured in months or years.
Medium (M)	Exploits are occasionally observed. Expected time-between-exploits for a vulnerable system is measured in days.
High (H)	Exploits are frequently observed. Expected time-between-exploits for a vulnerable system is measured in hours, minutes, or seconds.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is Medium.

2.2.2 General Remediation Level (GRL)

The General Remediation Level metric measures the availability of remediation measures that can mitigate the vulnerability other than changing the configuration setting or rendering it useless (e.g., disabling the affected feature, removing the software, installing a third-party security control that overrides the setting). Examples of remediation measures include running antivirus software on a system to restrict which emails a user can read and configuring a host-based firewall to block incoming network connection attempts from external hosts. Remediation measures do not have to be limited to those that are internal to the system—options such as using network-based security controls (e.g., network firewalls, network-based antivirus software) and training users on avoiding poor security practices may also be effective. The effectiveness of the available remediation measures determines the General Remediation Level score.

The possible values for this metric are listed in Table 8. The less effective the available remediation measures, the higher the score.

Table 8. General Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are available to mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 76% and 100%. (A decrease of 100% means that the available remediation measures are able to entirely prevent exploitation.)
Medium (M)	Remediation measures are available to partially mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 26% and 75%.
Low (L)	Remediation measures are available to slightly mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 1% and 25%.
None (N)	Remediation measures are not available.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is None.

2.3 Environmental Metrics

Differences between environments can have a large effect on the risk that a vulnerability poses to a particular organization and its stakeholders. The CMSS environmental metrics capture the characteristics of a vulnerability that are associated with a particular IT environment. Each organization computing CMSS metrics can determine an appropriate definition of IT environment, such as the entire enterprise or a logical or physical subset of the enterprise (e.g., a facility, a division, a small group of related systems). Since environmental metrics are optional, each includes a metric value that has no effect on the score. This value is used when the scoring analyst feels the particular metric does not apply and wishes to ignore it.

The environmental metrics customize the previously computed base and temporal metrics. The environmental metrics measure three aspects of vulnerability severity: Local Exploit Level, Local Remediation Level, and Local Impact. Similar to the General Exploit Level and General Remediation Level temporal metrics, the Local Exploit Level and Local Remediation Level environmental metrics produce a scaling factor that is applied to the Exploitability components of the base metric. The Local Impact environmental metric comprises several metrics that adjust the base impact metrics to take environment-specific characteristics into account.

The environmental metrics are intended to measure deviations from the assumptions about environments that were used to compute the base and temporal metrics. Therefore, environmental metrics should be scored relative to those assumptions.

2.3.1 Local Exploit Level

The local exploit level can be captured using two environmental metrics: Local Vulnerability Prevalence and Perceived Target Value.

2.3.1.1 Local Vulnerability Prevalence (LVP)

The Local Vulnerability Prevalence metric measures the prevalence of vulnerable systems in a specific environment. It is intended to approximate the percentage of systems that could be affected by the vulnerability. The Local Vulnerability Prevalence depends both on the prevalence of the configuration issue under scrutiny and the prevalence of its exploitation. For vulnerabilities dependent on user actions, the prevalence of exploitation depends on the probability that users in this environment will perform the actions required for vulnerability exploitation. For example, if 80% of the systems contain a particular

vulnerability but only half of the users of those systems are expected to engage in the improper actions that could permit the vulnerability to be exploited, then 40% of the total environment is at risk. Thus, the Local Vulnerability Prevalence would be rated as medium. The Local Vulnerability Prevalence also takes into account, when appropriate, how frequently the vulnerability is relevant for targets, such as how often the vulnerable software is run, how many hours per day the vulnerable software is running, and how much usage exposes the software to threats (for example, how many web sites or emails a user accesses). The possible values for this metric are listed in Table 9. The greater the approximate percentage of vulnerable systems, the higher the score.

Table 9. Local Vulnerability Prevalence Scoring Evaluation

Metric Value	Description
None (N)	No target systems exist, or targets are so highly specialized that they only exist in a laboratory setting. Effectively 0% of the environment is at risk.
Low (L)	Targets exist inside the environment, but on a small scale. Between 1% and 25% of the total environment is at risk.
Medium (M)	Targets exist inside the environment, but on a medium scale. Between 26% and 75% of the total environment is at risk.
High (H)	Targets exist inside the environment on a large scale. Between 76% and 100% of the total environment is considered at risk.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is Medium.

2.3.1.2 Perceived Target Value (PTV)

The Perceived Target Value metric measures the likelihood of attack using the configuration issue in an environment relative to vulnerable systems in other environments. The metric indicates the level of motivation for an attacker to attempt to exploit the vulnerability in the environment relative to other environments. The possible values for this metric are listed in Table 10. The higher the Perceived Target Value, the higher the score.

Table 10. Perceived Target Value Scoring Evaluation

Metric Value	Description
Low (L)	The targets in this environment are perceived as low value by attackers. Attackers have low motivation to attack the target system relative to other systems with the same vulnerability.
Medium (M)	The targets in this environment are perceived as medium value by attackers. Attackers are equally motivated to attack the target system and other systems with the same vulnerability.
High (H)	The targets in this environment are perceived as high value by attackers. Attackers are highly motivated to attack the target system relative to other systems with the same vulnerability.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is Medium.

2.3.2 Local Remediation Level (LRL)

The Local Remediation Level metric measures the level of protection against a vulnerability within the local IT environment and captures both how widespread mitigation implementation is and how effective such mitigation is. To calculate the environmental score, the Local Remediation Level metric replaces the temporal General Remediation Level metric, which measures only the availability of remediation measures, not the implementation.

The possible values for this metric are listed in Table 11. The less thorough or effective the implementation of remediation measures, the higher the score.

Table 11. Local Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are implemented to mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 76% and 100%. (A decrease of 100% means that the implemented remediation measures entirely prevent exploitation.)
Medium (M)	Remediation measures are implemented to partially mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 26% and 75%.
Low (L)	Remediation measures are implemented to slightly mitigate the vulnerability in such a way as to collectively decrease the incidence of exploitation by between 1% and 25%.
None (N)	Remediation measures are not implemented.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is None.

2.3.3 Local Impact

The Local Impact can be captured using seven environmental metrics. The first three—Environment Confidentiality Impact, Environment Integrity Impact, and Environment Availability Impact—take the place of the corresponding base impact metrics (Confidentiality Impact, Integrity Impact, and Availability Impact). The fourth, Collateral Damage Potential, augments the three Environment Impact metrics. The remaining three environmental metrics (Confidentiality Requirement, Integrity Requirement, and Availability Requirement) are used to compute scaling factors that are applied to the three Environment Impact metrics.

2.3.3.1 Environment Confidentiality, Integrity, and Availability Impact (EC, EI, EA)

The Environment Confidentiality, Integrity, and Availability Impact metrics enable the analyst to customize the environmental score if the privileges in the environment differ significantly from best practices related to the vulnerability. For example, suppose that a vulnerability permitted an attacker to gain unauthorized access to a system with the user's privileges. In the base impact metrics, this would have been recorded as a partial impact to confidentiality, integrity, and availability, under the assumption that the user was running with limited user privileges. However, if a particular environment permitted users to run with full, root-level privileges, then this could be taken into account through the Environment Impact metrics.

The Environment Impact metrics include all the same definitions as the base impact metrics; each can be set to None, Partial, or Complete. Additionally, each Environment Impact metric also includes a Not Defined (ND) value. The definition of ND is: "Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is the value assigned to the corresponding base impact metric."

2.3.3.2 Collateral Damage Potential (CDP)

The Collateral Damage Potential metric measures the potential for loss of life or physical assets through damage or theft of property or equipment. The metric may also measure economic loss of productivity or revenue. This metric can adjust the local impact score to account for application importance. For example, a vulnerability that permits an attacker to gain user-level access to an application (e.g., DNS server, database server) can be scored differently on a system that uses the application in a trivial way versus another system that uses the application in a critical way. The possible values for this metric are listed in

Table 12. The greater the damage potential, the higher the vulnerability score. Each organization must determine for itself the precise meaning of "slight," "moderate," "significant," and "catastrophic" in the organization's environment.

Table 12. Collateral Damage Potential Scoring Evaluation

Metric Value	Description
None (N)	There is no potential for loss of life, physical assets, productivity or revenue.
Low (L)	Successful exploitation of this vulnerability may result in slight physical or property damage or loss. Or, there may be a slight loss of revenue or productivity.
Low-Medium (LM)	Successful exploitation of this vulnerability may result in moderate physical or property damage or loss. Or, there may be a moderate loss of revenue or productivity.
Medium-High (MH)	Successful exploitation of this vulnerability may result in significant physical or property damage or loss. Or, there may be a significant loss of revenue or productivity.
High (H)	Successful exploitation of this vulnerability may result in catastrophic physical or property damage or loss. Or, there may be a catastrophic loss of revenue or productivity.
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is None.

2.3.3.3 Confidentiality, Integrity, Availability Requirements (CR, IR, AR)

The Confidentiality Requirement, Integrity Requirement, and Availability Requirement metrics enable the analyst to customize the environmental score depending on the importance of the affected IT asset to an organization, measured in terms of confidentiality, integrity, and availability. That is, if an IT asset supports a business function for which availability is most important, the analyst can assign a greater value to availability, relative to confidentiality and integrity. Each security requirement has three possible values: "Low," "Medium," or "High," with "Medium" being the default.

The full effect on the environmental score is determined by the corresponding Environment Impact metrics. That is, these metrics modify the environmental score by reweighting the Environment Confidentiality, Integrity, and Availability Impact metrics. For example, the Environment Confidentiality Impact metric has *increased* weight if the Confidentiality Requirement is "High." Likewise, the Environment Confidentiality Impact metric has *decreased* weight if the Confidentiality Requirement is "Low." The Environment Confidentiality Impact metric weighting is neutral if the Confidentiality Requirement is "Medium." This same logic is applied to the Environment Integrity and Availability Requirements.

Note that the Confidentiality Requirement will not affect the environmental score if the Environment Confidentiality Impact is set to "None." Also, increasing the Confidentiality Requirement from "Medium" to "High" will not change the environmental score when all of the Environment Impact metrics are set to "Complete" because the Impact subscore (the part of the score that calculates impact) is already at a maximum value of 10.

The possible values for the security requirements are listed in Table 13. For brevity, the same table is used for all three metrics. The greater the security requirement, the higher the score.

In many organizations, IT resources are labeled with criticality ratings based on network location, business function, and potential for loss of revenue or life. For example, the U.S. government assigns every unclassified IT asset to a grouping of assets called a System. Every System must be assigned three "potential impact" ratings to show the potential impact on the organization if the System is compromised

according to three security objectives: confidentiality, integrity, and availability. Thus, every unclassified IT asset in the U.S. government has a potential impact rating of low, moderate, or high with respect to the security objectives of confidentiality, integrity, and availability. This rating system is described within Federal Information Processing Standards (FIPS) 199.[12] CCSS follows this general model of FIPS 199, but does not require organizations to use a particular methodology for assigning the low, medium, and high impact ratings.

Table 13. Confidentiality, Integrity, and Availability Requirements Scoring Evaluation

Metric Value	Description
Low (L)	Loss of [confidentiality | integrity | availability] is likely to have only a limited adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).
Medium (M)	Loss of [confidentiality | integrity | availability] is likely to have a serious adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).
High (H)	Loss of [confidentiality | integrity | availability] is likely to have a catastrophic adverse effect on the organization or individuals associated with the organization (e.g., employees, customers).
Not Defined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric. The default value is Medium.

2.4 Base, Temporal, and Environmental Vectors

The CCSS vector facilitates its open nature. 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 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. If a temporal or environmental metric is not to be used, it is given a value of "ND" (not defined). The base, temporal, and environmental vectors are shown below in Table 14.

Table 14. Base, Temporal, and Environmental Vectors

Metric Group	Vector
Base	AV:[L,A,N]/AC:[H,M,L]/Au:[M,S,N]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C]/PL:[R,U,A,ND]/EM:[A,P]
Temporal	GEL:[N,L,M,H,ND]/GRL:[H,M,L,N,ND]
Environmental	LVP:[N,L,M,H,ND]/PTV:[L,M,H,ND]/LRL:[N,L,M,H,ND]/EC:[N,P,C,ND]/EI:[N,P,C,ND]/ EA:[N,P,C,ND]/CDP:[N,L,LM,MH,H,ND]/CR:[L,M,H,ND]/IR:[L,M,H,ND]/AR:[L,M,H,ND]

For example, a vulnerability with base metric values of "Access Vector: Low, Access Complexity: Medium, Authentication: None, Confidentiality Impact: None, Integrity Impact: Partial, Availability Impact: Complete, Privilege Level: Not Defined, Exploitability Method: Active" would have the following base vector: "AV:L/AC:M/Au:N/C:N/I:P/A:C/PL:ND/EM:A." Temporal metric values of "General Exploit Level: Medium, General Remediation Level: Medium" would produce the temporal vector: "GEL:M/GRL:M." Environmental metric values of "Local Vulnerability Prevalence: High, Perceived Target Value: Medium, Local Remediation Level: Low, Environment Confidentiality Impact: Not Defined, Environment Integrity Impact: Full, Environment Availability Impact: Not Defined,

[12] http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

Collateral Damage Potential: Not Defined, Confidentiality Requirement: Medium, Integrity Requirement: High, Availability Requirement: Low" would produce the following environmental vector: "LVP:H/PTV:M/LRL:L/EC:ND/EI:C/EA:ND/CDP:ND/CR:M/IR:H/AR:L."

3. Scoring

This section explains how CCSS scoring is performed. It provides guidelines on performing scoring, and it defines the equations used for base, temporal, and environmental score generation. Section 4 provides scoring examples to help illustrate the scoring process and the use of the equations.

3.1 Guidelines

Below are guidelines that should help analysts when scoring security configuration issues. These guidelines are intended primarily for analysts that are creating base scores, although they may be of interest to many others because of the insights they provide into the significance of the base scores and the assumptions made when performing scoring.

There are potentially multiple vectors and base scores for a single configuration issue, if there is more than one way in which it can be configured that causes a negative security impact. For example, some Windows platforms have an Application Management service, which 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 base 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 then the appropriate base vector and score would be selected for each situation based on the organization's policy and actual system settings.

3.1.1 General

SCORING TIP #1: Configuration issue scoring should not take into account any interaction with other configuration issues, software flaws, or vulnerabilities. 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 system 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 system if the Web server service is enabled, then analysts should assume the service is enabled.

SCORING TIP #2: When scoring the base metrics for a configuration issue, consider the direct impact to the target system only. For example, consider a misconfigured access control list that allows users to place files of their choice (such as malware) in a trusted network share: the impact to the systems of users that download and execute the malware could be much greater than the impact to the target system. However, this is an indirect impact, and should not be considered in base scoring.[13]

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

[13] 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.

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 farfetched—exceedingly 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 system could be set to grant one set of excess privileges to user A, grant a different set of excess privileges to user B, and grant insufficient privileges to user C. However, we cannot score endless 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 scoring for the configuration issues within its own systems would compare the requirements specified in its own policies and security configuration checklists against the systems' actual security settings to identify configuration discrepancies and then select the appropriate vector and score for each configuration issue.

3.1.2 Base Metrics

SCORING TIP #4: When a configuration issue can be exploited both locally and from the network, the "Network" value should be chosen for the Access Vector. 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 "Network" value should be chosen.

SCORING TIP #5: Many client applications and utilities 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 configuration issues caused these to ignore certain types of content that they should be examining, then analysts should score the Access Vector as "Network".

SCORING TIP #6: If the configuration issue exists in an authentication scheme itself (e.g., Pluggable Authentication Module [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 #7: Configuration issues that give root-level access should be scored with complete loss of confidentiality, integrity, and availability, while issues that give lesser degrees of access, such as user-level, 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 #8: Configuration issues that permit a partial or complete loss of integrity often also permit an impact to availability. For example, an attacker who is able to modify records can probably also delete them.

SCORING TIP #9: 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 as permitting a partial loss of integrity.

3.2 Equations

Scoring equations and algorithms for the CCSS base, temporal, and environmental metric groups are described below. Further information on the origin and testing of the original CVSS equations, which the CCSS equations are based on, is available at http://www.first.org/cvss/.

3.2.1 Base Equation

The base equation is the foundation of CCSS scoring. The CCSS base equation is identical to the CVSS base equation:

```
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 * Authentication * AccessComplexity

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

AccessVector = case AccessVector of
                requires local access: 0.395
                adjacent network accessible: 0.646
                network accessible: 1.0
Authentication = case Authentication of
                requires multiple instances of authentication: 0.45
                requires single instance of authentication: 0.56
                requires no authentication: 0.704
AccessComplexity = case AccessComplexity of
                high: 0.35
                medium: 0.61
                low: 0.71

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
```

The base equation does not include two of the base metrics: Exploitability Method and Privilege Level. Exploitability Method is used in the definitions for some of the base metrics, so it is used in choosing the associated values. Privilege Level is used by analysts when calculating certain environmental metrics.

3.2.2 Temporal Equation

If employed, the temporal equation will combine the temporal metrics with the base metrics to produce a temporal score ranging from 0 to 10. Note that the Impact component is not changed from the base score. The temporal equation modifies the Exploitability component of the base equation:

```
TemporalScore = round_to_1_decimal(((0.6 * Impact) + (0.4 * TemporalExploitability) – 1.5) * f(Impact))

TemporalExploitability = min(10, Exploitability * GeneralExploitLevel * GeneralRemediationLevel)

GeneralExploitLevel = case GeneralExploitLevel of
                    none: 0.6
                    low: 0.8
                    medium: 1.0
                    high: 1.2
                    not defined: 1.0

GeneralRemediationLevel = case GeneralRemediationLevel of
                    none: 1.0
                    low: 0.8
                    medium: 0.6
                    high: 0.4
                    not defined: 1.0
```

3.2.3 Environmental Equation

If employed, the environmental equation will combine the environmental metrics with the temporal and base metrics to produce an environmental score ranging from 0 to 10. The temporal GeneralExploitLevel metric is included in the environmental equation; however, the temporal GeneralRemediationLevel metric is not, because it is replaced by the environmental LocalRemediationLevel metric. The temporal remediation metric examines *availability* of remediation measures; the environmental remediation metric examines the *implementation* of remediation measures in the local environment. The environmental equation is computed using the following:

```
EnvironmentalScore = round_to_1_decimal(((0.6 * EnvironmentalImpact) + (0.4 * EnvironmentalExploitability) –1.5) *
f(Impact))

EnvironmentalImpact = min(10, 10.41*(1 - (1 - EnvConfImpact * ConfReq) * (1 - EnvIntegImpact * IntegReq) *
(1 - EnvAvailImpact * AvailReq)) * CollateralDamagePotential)

EnvironmentalExploitability = min(10, Exploitability * GeneralExploitLevel * LocalExploitLevel *
LocalRemediationLevel)

LocalExploitLevel = LocalVulnerabilityPrevalence * PerceivedTargetValue

EnvConfImpact = case EnvironmentConfidentialityImpact of
                    none: 0.0
                    partial: 0.275
                    complete: 0.660
                    not defined: ConfImpact
```

```
EnvIntegImpact = case EnvironmentIntegrityImpact of
                    none: 0.0
                    partial: 0.275
                    complete: 0.660
                    not defined: IntegImpact
EnvAvailImpact = case EnvironmentAvailabilityImpact of
                    none: 0.0
                    partial: 0.275
                    complete: 0.660
                    not defined: AvailImpact

ConfReq = case ConfReq of
                    low: 0.5
                    medium: 1.0
                    high: 1.51
                    not defined: 1.0
IntegReq = case IntegReq of
                    low: 0.5
                    medium: 1.0
                    high: 1.51
                    not defined: 1.0
AvailReq = case AvailReq of
                    low: 0.5
                    medium: 1.0
                    high: 1.51
                    not defined: 1.0

CollateralDamagePotential = case CollateralDamagePotential of
                    none: 1.0
                    low: 1.25
                    low-medium: 1.5
                    medium-high: 1.75
                    high: 2.0
                    not defined: 1.0

LocalVulnerabilityPrevalence = case LocalVulnerabilityPrevalence of
                    none: 0.6
                    low: 0.8
                    medium: 1.0
                    high: 1.2
                    not defined: 1.0

PerceivedTargetValue = case PerceivedTargetValue of
                    low: 0.8
                    medium: 1.0
                    high: 1.2
                    not defined: 1.0

LocalRemediationLevel = case LocalRemediationLevel of
                    none: 1.0
                    low: 0.8
                    medium: 0.6
                    high: 0.4
                    not defined: 1.0
```

4. Scoring Examples

This section provides examples of how CCSS would be used for several types of configuration issues.[14] The issues in the examples are from the Common Configuration Enumeration (CCE) version 5 specification[15], which assigns unique identifiers to security configuration issues for operating systems and applications. The examples focus on base metrics, but the last one also includes temporal and environmental metrics.

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 the events automatically fail to be logged for all users, the Exploitation Method is "Passive" and the Access Complexity is "Low." Some of the types of events logged in kernel level auditing are remotely triggered, so the Access Vector is "Network". 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 Privilege Level is "Not Defined."

The base vector for this weakness is AV:N/AC:L/Au:N/C:N/I:P/A:N/PL:ND/EM:P. This vector produces an impact subscore of 2.9, an exploitability subscore of 10.0, and 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 Exploitation Method is "Passive" because an authorized function is not available, and the Access Complexity is "Low.". The impact is rated as a partial compromise of availability, because the functionality is not available, and no compromise of confidentiality or integrity. The Privilege Level is "Not Defined." The base vector for case 1 is AV:L/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:P. This vector produces an impact subscore of 2.9, an exploitability subscore of 3.9, and a base score of 2.1.

---

[14] The base 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.

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

For case 2, the Exploitation Method is “Active” because the user has to act to attempt to modify the file. 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 Privilege Level is “Not Defined.” The Access Complexity is “Low” 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/PL:ND/EM:A. This vector produces an impact subscore of 6.4, an exploitability subscore of 3.9, and 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 Exploitation Method is “Active” because the user has to act to create a pagefile. 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 system availability, because having small pagefiles or no pagefile could seriously impact system performance. There is no impact on confidentiality or integrity. The Privilege Level is “Not Defined.”

The base vector for this weakness is: AV:L/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:A. This vector produces an impact subscore of 2.9, an exploitability subscore of 3.9, and 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 systems, 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”. Because the security concern is attackers guessing passwords, the Exploitation Method is “Active.” 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 system confidentiality. There is no impact on integrity or availability because further impact on the system would be secondary effects that occur once the attacker is actually able to log onto it. The Privilege Level is “Not Defined.” The base vector is: AV:L/AC:H/Au:N/C:P/I:N/A:N/PL:ND/EM:A. This vector produces an impact subscore of 2.9, an exploitability subscore of 1.9, and a base score of 1.2.

For the second case (longer-than-recommended lockout), the Access Vector is “Local”. Because the security concern is users not being able to access their accounts, the Exploitation Method is “Passive” and the Access Complexity is “Low.” 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 system availability. There is no impact on confidentiality or integrity. The Privilege Level is “Not Defined.” The base vector is: AV:L/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:P. This vector produces an impact subscore of 2.9, an exploitability subscore of 3.9, and 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 system, but should not. Access to this file requires a local account on the computer, so the Access Vector is “Local”. The Exploitation Method is “Active” because the user has to act to shut down the system. The Access Complexity is “Low” because the user can simply use regular OS features to shut down the system. 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 system availability (the system can be shut down by users at will). There is no impact on confidentiality or integrity. The Privilege Level is “Not Defined.” The base vector for this weakness is: AV:L/AC:L/Au:N/C:N/I:N/A:C/PL:ND/EM:A. This vector produces an impact subscore of 6.9, an exploitability subscore of 3.9, and a base score of 4.9.

Another case with possible security implications is if no users, including administrators, had the right to shut the system down. This could delay administrators in shutting down the system when needed, so it would be a partial compromise of system availability, with no impact on confidentiality or integrity. The Privilege Level is “Not Defined.” The Exploitation Method is “Passive” and the Access Complexity is “Low” because it prevents an authorized action from occurring. As with the first case, the Access Vector is “Local” and the Authentication Metric is “None.” The base vector would be AV:L/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:A. This vector produces an impact subscore of 2.9, an exploitability subscore of 3.9, and 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 system about the local system’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 system, 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 system, an operational issue, not a security issue.

For this case, the Access Vector is “Network” because remote Web sites can access the usage data. Because a remote web site must act to get the data, the Exploitation Method is “Active”. 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 Privilege Level is “Not Defined.”

The base vector would be AV:N/AC:H/Au:N/C:P/I:N/A:N/PL:ND/EM:A. This vector produces an impact subscore of 2.9, an exploitability subscore of 4.9, and 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 only. The Privilege Level is "Not Defined." No special action is needed, so the Exploitation Method is "Passive" and the Access Complexity is "Low." The Access Vector would be "Network" since these are remote sessions to the target. Authentication would be "None". The base vector would be AV:N/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:P. This vector produces an impact subscore of 2.9, an exploitability subscore of 10.0, and 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 system 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, but this is inconsequential when compared to the other implications of having the setting too high. 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, with the Privilege Level "User Level." The Access Vector would have to be "Network" since this is a remote session to the target. The Access Complexity would be "High" and the Exploitation Method is "Active" because the attacker would first have to gain access to the remote system, 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/PL:U/EM:A. This vector produces an impact subscore of 6.4, an exploitability subscore of 4.9, and 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 systems 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 system integrity. 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 systems, causing a partial impact on availability. The Privilege Level is "Not Defined."

Because remote systems could gain unauthorized access, the Access Vector is "Network". The Exploitation Method is "Active" and the Access Complexity would be "Low" because the attacker would 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/PL:ND/EM:A. This vector produces an impact subscore of 4.9, an exploitability subscore of 10.0, and 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. The Exploitation Method would be "Passive" because it prevents authorized actions. If this service is enabled but should be disabled, it allows a user to install or remove programs, which could alter the system's integrity. The Exploitation method would be "Active."

Both cases have an Access Vector of "Local", Access Complexity of "Low", Authentication of "None", and Privilege Level of "Not Defined."

The base vector for the first case is AV:L/AC:L/Au:N/C:N/I:N/A:P/PL:ND/EM:A, with an impact subscore of 2.9, an exploitability subscore of 3.9, and 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/PL:ND/EM:P, also with an impact subscore of 2.9, an exploitability subscore of 3.9, and 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 system may not be able to get an IP address, thus preventing use of IP services—partial impact on availability, with a Privilege Level of "Not Defined" and an Exploitation Method of "Passive" because authorized services cannot be used. If the DHCP client is enabled but should be disabled, then the system may be able to get access to IP services that it should not be able to—a partial impact on confidentiality—although arguably a system could just set its own IP address and get access to IP services with or without a DHCP client. The Privilege Level would be "Not Defined" and the Exploitation Method "Active."

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/PL:ND/EM:P, with an impact subscore of 2.9, an exploitability subscore of 3.9, and 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/PL:ND/EM:A, also with an impact subscore of 2.9, an exploitability subscore of 3.9, and 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 systems) also perform packet filtering (firewall capabilities). So if the service is disabled but should be enabled, it could expose the system's network services to unauthorized access from other systems (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 Privilege Level would be "Not Defined."

The Access Vector would be "Network" because other systems could potentially access network services, other systems' network traffic might go unprotected, and other systems might be unable to communicate

with the system (e.g., if IPsec is required for communications but unavailable). The Exploitation Method is "Passive" and 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/PL:ND/EM:P. This vector produces an impact subscore of 6.4, an exploitability subscore of 10.0, and a base score of 7.5.

4.12 CCE-2776-3

CCE-2776-3 in Windows XP is: "Automatic Logon should be properly configured." If Automatic Logon is permitted, then anyone with physical access to the computer could boot it and be logged on with the user's stored credentials, thus gaining unauthorized access as that user. This partially impacts confidentiality, integrity, and availability, assuming that the credentials grant user-level access. The Privilege Level would be set to "User Level."

The Exploitability Method would be "Active" because an attacker would have to physically access and boot the computer to gain unauthorized access as the user. The Access Vector would be "Local" because physical access to the computer is required. Access Complexity is "Low" because booting the computer is trivial if local access has already been achieved, and Authentication is "None."

The base vector would be AV:L/AC:L/Au:N/C:P/I:P/A:P/PL:U/EM:A. This vector produces an impact subscore of 6.4, an exploitability subscore of 3.9, and a base score of 4.6.

The following provides an example of possible temporal and environmental metrics for this configuration issue. There are two temporal metrics: General Exploit Level and General Remediation Level. The General Exploit Level would be set to "Low" if exploits of this configuration issue are rarely observed. The General Remediation Level would take into account available remediation measures other than changing the configuration setting, such as physically securing the computer. These measures could partially mitigate the threats, so the General Remediation Level would be set to "Medium." The temporal vector would be GEL:L/GRL:M. This produces a temporal exploitability subscore of 1.9 and a temporal score of 3.7. (There is not a temporal impact subscore because the temporal metrics do not change the base impact subscore.)

There are several environmental metrics to determine, and this example for those metrics is based on a hypothetical local environment. The Local Exploit Level is based on the Local Vulnerability Prevalence and the Perceived Target Value. In the example environment, the prevalence of the configuration is low, but the vulnerability can be exploited at any time. Still, the overall prevalence is low, so the Local Vulnerability Prevalence is set to "Low." The Perceived Target Value is considered "Low" because attackers with physical access to the facilities and equipment are more likely to be interested in other systems.

The next environmental metric is the Local Remediation Level. In the example environment, there is limited physical security for the systems but no other remediation strategies deployed. So the Local Remediation Level is set to "Low."

The final group of environment metrics is the Local Impact metrics. In the example environment, most users run with root-level privileges instead of user-level privileges. Because the Privilege Level is set to "User Level", indicating that an attacker can gain the access level of the user, this means that an attacker can gain root-level access through this configuration issue. So the Environment Confidentiality, Integrity, and Availability Impact metrics are each set to "Complete" to indicate the increased level of access. The Collateral Damage Potential is set to "Low" because exploitation would cause only slight damage and

loss. In the example environment, the need for availability for the targets is low as compared to preserving their confidentiality and integrity, so the Availability Requirement is set to “Low” and the Confidentiality and Integrity Requirements are set to “Medium.”

The environmental vector would be LVP:L/PTV:L/LRL:L/EC:C/EI:C/EA:C/CDP:L/CR:M/IR:M/AR:L. The environmental exploitability subscore is 1.6 and the impact subscore is 10.0, for an exploitability score of 6.1.

5. Comparing CCSS to CVSS and CMSS

CCSS is based on CVSS and CMSS, so there are many similarities among the three specifications. However, there are some important differences as well. This section provides a brief discussion of the major differences between the specifications. Individuals interested in more details on the differences are encouraged to compare the specifications side-by-side. The specifications have similar structures, making such comparisons easy.[16]

For the base metrics, all three specifications use the same six metrics and the same equations for calculating scores. The descriptions for each metric have been adjusted to fit the characteristics of the category of vulnerabilities that they cover. The most notable difference is that CCSS also measures the type of exploitation: active or passive. Active exploitation refers to an attacker performing actions to take advantage of a weakness, while passive exploitation refers to vulnerabilities that prevent authorized actions from occurring, such as a configuration setting that prevents audit log records from being generated for security events. The Exploitability base metrics in CCSS are defined differently for active and passive exploitation because of the differences in the ease of exploitation.

The temporal and environmental components of the three specifications are quite different. The temporal and environmental components of CCSS and CMSS are based on those from CVSS, but have major differences. The temporal metrics in CVSS measure the availability of exploit code, the level of available remediations for the software flaw (e.g., patches), and the confidence in the existence of the vulnerability. These are not relevant for the types of vulnerabilities addressed by CCSS and CMSS, because their vulnerabilities can be exploited without exploit code and are already known to exist. Also, CMSS vulnerabilities and many CCSS vulnerabilities do not have complete remediations. So CCSS and CMSS have similar sets of temporal metrics, quite different from those of CVSS, that address the general prevalence of attacks against the vulnerability and the general effectiveness of available remediation measures, such as using antivirus software or conducting awareness activities.

CCSS and CMSS also offer similar sets of environmental metrics, which are considerably more complex than CVSS's metrics. CVSS has three: Collateral Damage Potential, Target Distribution, and Security Requirements. These metrics are all part of CCSS and CMSS as well, although Target Distribution has been renamed Local Vulnerability Prevalence. Two other metrics have been added to CCSS and CMSS: Perceived Target Value, which measures how attackers value the targets in a specific local environment as opposed to other environments, and Local Remediation Level, which measures the effectiveness of mitigation measures in the local environment. CCSS and CMSS also divide their environmental metrics into two groups: Exploitability and Impact. This allows Exploitability and Impact environmental subscores to be generated for CCSS and CMSS; such subscores are not available in CVSS.

[16] The other specifications are NIST IR 7435 and draft NIST IR 7517 (http://csrc.nist.gov/publications/PubsNISTIRs.html).

6. Conclusions and Future Work

The Common Configuration Scoring System (CCSS) is an open, standardized method for measuring and scoring software security configuration issue vulnerabilities. CCSS is closely related to the Common Vulnerability Scoring System (CVSS) and the Common Misuse Scoring System (CCSS), methods to measure and score software flaws and software feature misuse vulnerabilities, respectively. These three standardized vulnerability measurement and scoring methods enable the comparison of analyses performed by different people and different companies over time.

By using CCSS, organizations can better analyze their risk caused by security configuration issues. The measures and scores produced from CCSS enable more informed security decisions that consider the trade-off between functionality and exposure to vulnerabilities. Because it is standardized, CCSS allows analysts to compare a system's security configuration to a baseline configuration and, as part of that analysis, use the CCSS scores for the settings that deviate from the baseline configuration as part of determining how significantly the system's security deviates from the expected state. Software vendors can use CCSS to communicate to their customers the relative security implications of using the various security configuration options in their products.

Because base measures and scores should be identical across all organizations, the intention is for the base and temporal data 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 and temporal measures and scores should not need to calculate them, and given the complexity and difficulty of performing base and temporal measurement and scoring, it is anticipated that few organizations will choose to expend the time and resources needed to do it themselves. However, environmental scoring is organization-specific, so it will need to be performed by individual organizations.

There is still significant work to be done before CCSS is ready for organizations to adopt. Base and temporal measures and scores need to be assigned to each security configuration issue in major software products and shared with the security community. This data can be used in conjunction with the CVSS and CMSS measures as a consistent set of measures for system vulnerabilities. In turn, this provides opportunities for using the data in threat models, risk assessments, and other security analysis activities. However, a way will need to be developed to relate data on various vulnerabilities to each other—there are many dependencies among vulnerabilities that affect their exploitability and impact. For example, one vulnerability might only be exploitable if a second vulnerability is also present or if a second vulnerability can grant user-level access. These dependencies need to be captured in a standardized way to facilitate the data's use for security modeling and analysis. Also, as of this writing, a mechanism has not yet been developed for measuring the security characteristics of a system using CVSS, CCSS, and CMSS together. The relative severity of individual vulnerabilities can be compared to each other, but the measures cannot be combined to measure the security of the system. Work to achieve such system-level measures is ongoing.

Organizations interested in assisting with the remaining CCSS work and related efforts should contact NIST so that research can be coordinated and duplication of efforts can be avoided.

7. Appendix A—Additional Resources

The following are resources related to CCSS.

- 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 equations. 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/.
- The CMSS draft specification, NISTIR 7517, *The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities*, is available at http://csrc.nist.gov/publications/PubsNISTIRs.html.
- 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.

8. Appendix B—Acronyms and Abbreviations

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

A Active
A Adjacent Network
A Availability
AC Access Complexity
AR Availability Requirement
AU Authentication
AV Access Vector

C Complete
C Confidentiality
CCE Common Configuration Enumeration
CCSS Common Configuration Scoring System
CDP Collateral Damage Potential
CMSS Common Misuse Scoring System
CR Confidentiality Requirement
CVSS Common Vulnerability Scoring System

DHCP Dynamic Host Configuration Protocol

EM Exploitability Method

FIPS Federal Information Processing Standards
FIRST Forum of Incident Response and Security Teams
FISMA Federal Information Security Management Act
FTP File Transfer Protocol

GEL General Exploit Level
GRL General Remediation Level

H High

I Integrity
IP Internet Protocol
IPsec Internet Protocol Security
IR Integrity Requirement
IR Interagency Report
IT Information Technology
ITL Information Technology Laboratory

L Local
L Low
LAN Local Area Network
LM Low-Medium
LRL Local Remediation Level
LVP Local Vulnerability Prevalence

M Medium

M	Multiple
MH	Medium-High
N	Network
N	None
ND	Not Defined
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency Report
NVD	National Vulnerability Database
OMB	Office of Management and Budget
OS	Operating System
P	Partial
P	Passive
PAM	Pluggable Authentication Module
PL	Privilege Level
PTV	Perceived Target Value
S	Single
SMTP	Simple Mail Transfer Protocol
USB	Universal Serial Bus