Vulnerability Scoring for Security Configuration Settings

Karen Scarfone
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-8930
+1-301-975-8136
karen.scarfone@nist.gov

Peter Mell
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-8930
+1-301-975-5572
mell@nist.gov

ABSTRACT
The best-known vulnerability scoring standard, the Common Vulnerability Scoring System (CVSS), is designed to quantify the severity of security-related software flaw vulnerabilities. This paper describes our efforts to determine if CVSS could be adapted for use with a different type of vulnerability: security configuration settings. We have identified significant differences in scoring configuration settings and software flaws and have proposed methods for accommodating those differences. We also generated scores for 187 configuration settings to evaluate the new specification.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *product metrics.*

General Terms
Experimentation, Management, Measurement, Security.

Keywords
Common Configuration Scoring System (CCSS), Common Vulnerability Scoring System (CVSS), risk assessment, security configuration, vulnerability, vulnerability scoring.

1. INTRODUCTION
The Common Vulnerability Scoring System (CVSS) is an open specification for measuring the major characteristics of security-related software flaws and scoring the potential impact of exploiting software flaws and the relative difficulty of exploitation [2]. CVSS is maintained by the CVSS Special Interest Group (CVSS-SIG) within the Forum for Incident Response and Security Teams (FIRST). CVSS has been widely adopted by the information technology (IT) community [1].

The original motivation for developing CVSS was to provide a consistent way of expressing vulnerability-related information that organizations could use to prioritize their mitigation responses to new software flaws. (Although the CVSS specification does not explicitly state that it only applies to software flaw vulnerabilities, every example in the specification involves a software flaw, and there are many other statements that strongly imply that its scope is limited to software flaws.) Since CVSS's initial development, we (and others) have wondered if CVSS could be extended for other purposes. For example, could it be applied to other types of vulnerabilities, such as security configuration settings? CVSS measures and scores for configuration settings would not be useful for mitigation prioritization, but could be quite valuable as inputs to quantitative risk assessment frameworks, threat models, and attack graphs and trees [6].

This paper describes our efforts to determine if CVSS version 2 (v2) can be adapted for measuring and scoring security configuration settings. We developed a modified version of the CVSS v2 specification and tested it on 187 entries from version 4.0 of the Common Configuration Enumeration (CCE) dictionary. We made additional changes to the specification based on the testing and feedback solicited from the CVSS-SIG and others. The updated specification has undergone review by the CVSS-SIG and was released for public review and comment in May 2008 [7].

Section 2 provides background on CVSS. Section 3 describes our identification of differences in scoring software flaws and security configuration settings, including our testing of the draft specification. Section 4 explains additional changes we made to the specification as a result of reviewer feedback. Section 5 presents several examples of scoring using the specification. Section 6 provides conclusions for the paper and proposes future work.

2. BACKGROUND
2.1 CVSS
CVSS base metrics are vulnerability attributes that are constant over time and across all implementations and environments. A formula is applied to the base metrics' values for a vulnerability to calculate its base score. Other CVSS metrics represent vulnerability attributes that change over time (temporal) and that are organization and implementation-specific (environmental).

The focus of our research is the base metrics. CVSS v2 has six base metrics, three of which relate to exploitability. AccessVector measures the exploitation range (e.g., local, over a network). Authentication measures whether an attacker must authenticate to a target before exploiting a vulnerability. AccessComplexity measures how hard it is to exploit a vulnerability after the target is

QoP'08, October 27, 2008, Alexandria, Virginia, USA.
ACM 978-1-60558-321-1/08/10.

accessed and any necessary authentication has been performed. Together, the exploitability metrics measure how readily an attacker can attempt to exploit a vulnerability. CVSS v2 also has three metrics related to impact. ConfImpact measures the potential degree of impact to a target's confidentiality, and IntegImpact and AvailImpact perform similar measurements for integrity and availability. The impact metrics measure the impact that an attacker can cause to a target by exploiting a vulnerability.

2.2 Vulnerability Dictionaries

The MITRE Corporation maintains dictionaries for publicly announced vulnerabilities in operating systems and applications. One of these dictionaries, Common Vulnerabilities and Exposures (CVE) [4], covers software flaws, and another dictionary, Common Configuration Enumeration (CCE) [3], addresses security configuration issues. When our work on adapting CVSS for security configuration issues began in July 2007, the current version of CCE was 4.0, and it comprised a single list of 910 configuration entries for several versions of Microsoft Windows, Microsoft Internet Explorer 7, and Microsoft Office 2007. Examples of CCE version 4.0 entries are:

- CCE-25: The required auditing for %SystemDrive% directory should be enabled
- CCE-100: The "minimum password length" policy should meet minimum requirements
- CCE-931: The "back up files and directories" user right should be assigned to the correct accounts

Note that CCE entries do not specify how each configuration should be set, only that each should be set properly. The implication is that users of CCE entries are responsible for determining what the proper settings should be for their environment.

3. DEVELOPING THE SPECIFICATION

When creating our new specification, we wanted to change as little of the CVSS v2 specification [2] as possible. This meant keeping the CVSS v2 metrics, score values, and formulas, and rewording only the portions of the specification that required it. We also chose to focus on configuration issues as listed in the CCE version 4.0 dictionary. Although Windows was the only software in that dictionary, we were mindful of configuration settings in other operating systems as we developed the specification.

To start our work, we reviewed the CVSS v2 specification and extracted the parts needed for our specification: the base metric definitions and the scoring guidelines. We modified these parts to change their context from software flaws to configuration settings. We also documented several scoring examples. We then gave the text to CVSS analysts from the National Vulnerability Database (NVD) [5] for review. After answering their questions and making a few changes to the text, we asked the analysts to use it to score a variety of CCE entries for Windows. We largely allowed the analysts to choose entries to score but encouraged them to choose a representative sample, which they did. Types of settings scored included audit settings, file permissions, user privileges, account lockout policies, web browser usage data settings, session timeout policies, service enabling/disabling, and administrator privilege escalation. After an analyst had scored a CCE entry, we independently scored the same entry and compared the results. A discrepancy indicated either human error or a problem with the specification. Each discrepancy was discussed in detail and resolved. Ultimately, 187 of the 910 CCE version 4.0 entries were scored. We also reviewed the score assigned to each entry to ensure that it seemed reasonable when compared to the severity of software flaws receiving similar scores. The rest of this section describes the changes to the specification that were identified during the scoring testing.

3.1 Base Metric Definitions

We looked for changes that might be needed in the base metric definitions. We found that some of the CCE entries were not clearly covered by the metric definitions; unlike software flaws, which permit attackers to take unauthorized actions against a host, these configuration settings prevented authorized actions. For example, if a user lacks the privileges needed to perform an action, then the host's availability for that user is negatively impacted.

To address such cases, we added a second class of configuration settings to the specification. The original class was for exploitable settings, such as excess privileges, unnecessary services running, and weak password policies. The new class was for settings that prevented authorized actions: insufficient privileges, unable to run needed services, lack of auditing, etc. We updated the AccessVector definition to include both classes of settings for all its possible values. We also updated the AccessComplexity definition so that all settings that prevent authorized actions are set to Low because they automatically affect the host—for example, auditing being disabled affects the host at all times and does not require attacker action. We determined that the Authentication metric needed no changes because it already covered both classes of settings.

We also expanded the impact metric definitions. For ConfImpact, we added "unauthorized access to the system" to the existing "information disclosure" definitions to make it clear that it should include unauthorized resource use. For IntegImpact, we added alterations to the system's configuration, such as installing unauthorized programs. We did not change the AvailImpact definitions because they already applied to both classes of settings.

3.2 Scoring Guidelines

We determined that a major change was needed to the scoring guidelines. This change involved addressing a key difference between software flaws and configuration settings: universality. A software flaw is an absolute: any organization would consider it undesirable. On the other hand, many security configuration settings are environment-specific, such as the number of minutes to wait before disconnecting an idle session, and do not necessarily have a "correct" value. In this example, decreasing the setting would make it less likely that an attacker could access and use an idle session, but the decrease would also reduce the availability of the session to the user. At its extreme, such as setting the idle time to one minute, the service might be highly unavailable to both users and attackers. At the other extreme, such as setting the idle time to one day, the service might be highly available to both users and attackers.

Because many settings do not have a “correct” value, scoring a setting often involves considering multiple possibilities. This is easy for settings with two possibilities (such as “enabled” and “disabled”), manageable for settings with three to five sequentially ordered possibilities (i.e., a clear progression from least secure to most secure), and far more complex for settings with more possibilities, such as an access control list (ACL) for a directory, which may have millions of possibilities. For example, CCE-411 is “The required permissions for the directory %SystemDrive% should be assigned”. An ACL for this directory could provide too many privileges to some users, too few privileges to others, and the appropriate privileges to yet other users, and the extent of the incorrect privileges could differ among the user accounts.

Because we wanted to try having a single score for each CCE (to be consistent with having a single score for each CVE), we initially devised a scheme for choosing which possible settings should be used to generate each score. If a configuration setting had two possible values with security implications, then both possibilities should be calculated (set to A but should be set to B; set to B but should be set to A) and the higher score chosen. If a configuration setting had three sequentially ordered values, then the low to high and high to low possibilities should be scored (set to 1 but should be set to 3; set to 3 but should be set to 1) and the higher score chosen. If a configuration setting had more complex possibilities, then the analyst should identify the broad cases that are most likely to occur and have security implications, score those, and choose the highest score. We also added an explanation that scores would “be generally representative of the relative importance of the configuration setting” and not what “would be assigned to configuration settings for a particular organization”, further explaining that organizations would need to generate their own scores based on their specific requirements.

4. REVISING THE SPECIFICATION

After the scoring test and the specification updates were completed, we sent the specification to the CVSS-SIG for review and received feedback from several members. The most common feedback was that having a single score for each CCE entry instead of multiple scores was of little value, and that having each organization generate its own custom scores would be inefficient and inconsistent. In response, we modified the specification so that there could be multiple scores for a single entry. When a configuration setting has a few possible values, analysts should consider the security implications of each combination of desired and actual settings and create a score for each combination. Sections 5.1 and 5.2 present examples of scoring settings with two possible values.

When a configuration setting has a larger number of possible values, analysts should generate a score for each common case. For example, if a timeout can be set to any number of seconds, then the analyst would consider the cases where the timeout is set too high and set too low. If setting the timeout to 0 disables it, then the analyst would also consider the case when the setting is disabled but should be enabled. An example of this is described in Section 5.3.

If a configuration setting can have multiple values simultaneously—such as an access control list that sets individual privileges for many users—then the analyst would consider the common cases and generate a separate score for each. A person assessing such a configuration setting would determine which cases applied to that setting, examine their base metrics’ values, choose the highest values from those metrics, and calculate a new score that encompasses all the applicable cases. Section 5.4 provides an example to illustrate this.

The next step in revising the specification was to redo its examples to have multiple scores where appropriate. We discovered that generating multiple scores for a configuration did not take significantly more time than generating a single score. To arrive at a single score, analysts often had to generate several scores and then choose the highest among them. When two or more cases had the same high score, analysts had to evaluate additional characteristics of the settings to determine which case should be selected. With the new procedure, the analyst no longer has to compare scores or do other evaluations to choose a single score. On the other hand, analysts may have to evaluate a few more cases than they otherwise would have. However, the analysts reported—and we observed during our own scoring—that the research conducted before scoring an entry generally takes considerably longer than generating all the scores. So having analysts generate multiple scores might not take substantially more time than generating single scores.

We also revised the specification to incorporate several examples of configuration settings from platforms other than Windows XP. In March 2008, CCE version 5.0 was released. Its entries are very similar to version 4.0 entries, with the main difference being that version 5.0 has a separate list of entries for each version of Windows (CCE version 4.0 had a single list for all versions of Windows), as well as lists for Internet Explorer 7, Office 2007, Red Hat Enterprise Linux 5, and Sun Solaris 10. We tested the specification on representative entries from these operating systems and applications to ensure that the specification was sufficiently flexible to accommodate them all, and that the specification produced scores that were reasonable approximations of the relative severity of each security configuration setting.

To clearly distinguish the new security configuration scoring specification from the CVSS v2 specification, we named the new specification the Common Configuration Scoring System (CCSS). The draft CCSS specification [7], which permits multiple scores for each configuration setting and includes examples for several platforms, has undergone review by the CVSS-SIG and others in the security community, and feedback on the specification has been positive.

5. SCORING EXAMPLES

Several examples of scoring using the draft CCSS specification are presented below. The security configuration settings in these examples have been selected to illustrate the range of complexity in settings, as well as a variety of platforms, and how the CCSS specification is flexible enough to accommodate these differences. The examples have been adapted from the draft CCSS specification [7] and from the CCE version 5.0 lists [3].

5.1 Example 1: One Option

CCE-4675-5 is a Sun Solaris 10 entry. Its CCE definition is “Kernel level auditing should be enabled or disabled as appropriate.” The definition indicates that there are two options for the setting. From a security standpoint, we are concerned about auditing being disabled when it should be enabled. (Arguably, in some cases there could be a performance impact if auditing is enabled when it should be disabled, but we consider this an operational issue and not a security issue.) If kernel level auditing is disabled, various security events will not be logged.

For the exploitation measures, the AccessVector is set to "Network" because security events to be logged could be generated from remote locations. The Authentication metric is set to "None" because no authentication is needed to generate security events. The AccessComplexity metric is set to "Low" because events fail to be logged by default, without any specific attacker action needed. For the impact measures, IntegImpact is set to "Partial" because the integrity of the host's security posture is somewhat degraded by the lack of auditing. ConfImpact and AvailImpact are set to "None".

These measures produce a base score of 5.0. This is comparable to the software flaw CVE-2004-1358, which is defined as "The patches (1) 114332-08 and (2) 114929-06 for Sun Solaris 9 disable the auditing functionality of the Basic Security Module (BSM), which allows attackers to avoid having their activity logged." The National Vulnerability Database assigned the same CVSS measures and score to CVE-2004-1358 [5] as the CCSS measures and score we assigned to CCE-4675-5.

5.2 Example 2: Two Options

CCE-3047-8 is a Windows XP entry, defined as "Application Management". It has two options: "enabled" and "disabled". If it is disabled, users cannot install applications; if it is enabled, users can install and uninstall applications. Both of these have potentially negative security implications.

For the exploitation measures, the setting only affects local users, so the AccessVector is set to "Local". No additional authentication is needed, so Authentication is "None". The AccessComplexity metric is "Low" because the setting is applied automatically, without any user action needed. For the impact measures, the impact might vary by case. In the first case (the service should be disabled but is not), users are unable to use the host to install new applications, thus impacting host availability (ConfImpact "None", IntegImpact "None", AvailImpact "Partial"). In the second case (the service should be enabled but is not), users can install and uninstall applications, which could affect the host's integrity and availability (ConfImpact "None", IntegImpact "Partial", AvailImpact "Partial").

The base score for the first case is 2.1, and the base score for the second case is 3.6. These scores are low on the base score scale, indicating that their severity is relatively minor. Users can only install and uninstall applications using their own privileges, so the assumption is that they have limited, user-level privileges and cannot install or remove system-level applications, which would merit a higher score. Also, the only people who can take advantage of the setting are local users who have already authenticated to Windows XP, thus limiting the opportunities for exploitation.

5.3 Example 3: Range of Options

CCE-2363-0 is a Windows Vista entry, with the definition "The 'account lockout duration' policy should meet minimum requirements". An account lockout occurs when too many consecutive failed authentication attempts happen. The lockout duration setting specifies how long (in minutes) the host should wait before accepting additional authentication attempts for the locked-out account. So for CCSS scoring purposes, we think of the possible improper settings as being too high or too low than a desirable value, such as that specified in an organization's policy or a vendor's security recommendations. If the setting's value is too high as compared to the desirable value, legitimate users will be unable to log onto the host for an extended period of time (partial impact to availability). The same is true if the value is set to 0, which keeps the account locked out until an administrator manually unlocks it. If the setting's value is too low, attackers will have more opportunities to guess the password (partial impact to confidentiality).

For cases where the lockout is too long (value is too high or set to 0), the AccessVector is set to "Local" because this involves local authentication. Authentication is "None" because no authentication is needed (for that matter, it is inherent in this scenario that authentication has been unsuccessful). AccessComplexity is "Low" because the lockout occurs automatically and it is easy for lockouts to occur during normal host use. The impact metrics are ConfImpact "None", IntegImpact "None", and AvailImpact "Partial". The base score for the too-long cases is 2.1.

For cases where the lockout is too short (value is too low), the AccessVector is set to "Local" because this involves local authentication. Authentication is "None" because no authentication is needed. AccessComplexity is "High" because the setting simply makes it slightly more likely that an attacker will guess a password. The impact would be to the confidentiality of passwords, so the impact metrics are ConfImpact "Partial", IntegImpact "None", and AvailImpact "None". The base score for the too-short cases is 1.2.

The scores for both types of cases are relatively low when compared to other settings. This is appropriate because an overly long lockout is primarily an inconvenience to users and has no permanent effect on the host, and a short lockout is unlikely on most hosts to lead to a password being guessed.

5.4 Example 4: Combination of Options

CCE-4693-8 is a Sun Solaris entry, defined as "File permissions for the /etc/cron.d/cron.allow file should be configured correctly". Users that are authorized to use cron are listed in this file. There are an essentially unlimited number of ways in which this CCE can be set incorrectly—in a worst-case situation, each user would have some privileges that they should not have and would be missing other privileges that they need. However, the main security issues with privileges can be summarized in four categories: 1) unauthorized users can modify the file (including deleting its contents), 2) unauthorized users can read the file, 3) authorized users cannot modify the file, and 4) authorized users cannot read the file.

For all four cases, the AccessVector is set to "Local" because a local user account is required. Authentication is "None" because no authentication is needed in addition to local OS authentication. AccessComplexity is "Low" because the user just needs to try to access the file.

The impact metrics vary among the cases. It is important to note that the metrics measure direct impact (to the file) and not indirect file (how changes made to the file could be used subsequently, such as authorized users that have been removed from the file listing no longer being able to use cron). For case 1, they are ConfImpact “None”, IntegImpact “Partial”, and AvailImpact “None”; the integrity of the file may be affected. For case 2, they are ConfImpact “Partial”, IntegImpact “None”, and AvailImpact “None”, because the list of cron users is exposed. For cases 3 and 4, they are ConfImpact “None”, IntegImpact “None”, and AvailImpact “Partial” because users are unable to perform functions for which they are authorized.

The base score for each case is 2.1. However, multiple cases could apply simultaneously to a single host, and even to a single user. For example, if a user who should have no access to the file has both read and modify rights, then both cases 1 and 2 would apply. That can be thought of as a single vulnerability, and the scores for the two cases would be combined by selecting the higher value from each of the metrics for the two cases and calculating a score from those values, in this case 3.6. Adding case 3 or 4 to cases 1 and 2, then selecting the highest values from each of the metrics for the cases generates a score of 4.6.

6. CONCLUSIONS AND FUTURE WORK

We have shown that it is feasible to adapt CVSS v2 for use with scoring security configuration settings while minimizing deviations from the CVSS specification. The major changes that we made were to expand the metric definitions to include settings that prevented authorized actions, and to permit multiple scores per configuration issue to reflect the possible combinations of desired and actual settings. The CVSS metric values and formulas were unchanged. Although our focus was on CVSS and CCEs, the differences that we have identified in scoring software flaws and configuration settings would be applicable to any method for performing quantitative assessments of host security, including threat models and attack graphs and trees.

We are currently finalizing the CCSS specification and are also beginning work on developing a similar specification for another class of vulnerabilities, software feature misuse. Software feature misuse is when an attacker takes advantage of the intended inherent functionality of software, not involving any software flaws or security configuration settings. Examples of this class of vulnerability include an attacker using social engineering to trick a user into opening a malicious email attachment or clicking on a malicious uniform resource locator (URL) in an email, and a malicious insider using secure shell (SSH) to transfer sensitive data files to an external system for fraudulent purposes. We intend for all of our vulnerability scoring specifications to be interoperable so that vulnerabilities can be measured and scored consistently regardless of vulnerability type.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge the assistance provided by the reviewers of the new specification, including Kurt Dillard, Robert Fritz, Tim Grance, Ron Gula, Dave Mann, Doug Noakes, Jim Ronayne, Murugiah Souppaya, Kim Watson, and the members of the CVSS-SIG. We are also particularly thankful for the contributions of Chuck Wergin and Dan Walsh, the NVD analysts who scored the CCE entries during testing and reviewed several specification drafts.

8. REFERENCES

[1] Forum of Incident Response and Security Teams. CVSS Adopters. http://www.first.org/cvss/eadopters.html

[2] Mell, P., Scarfone, K., and Romanosky, S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0. Forum of Incident Response and Security Teams, June 2007. http://www.first.org/cvss/cvss-guide.html

[3] MITRE Corporation. Common Configuration Enumeration (CCE). http://cce.mitre.org/

[4] MITRE Corporation. Common Vulnerabilities and Exposures (CVE). http://cve.mitre.org/

[5] National Institute of Standards and Technology. National Vulnerability Database. http://nvd.nist.gov/

[6] Scarfone, K. and Grance, T. A Framework for Measuring the Vulnerability of Hosts. In Proceedings of the 2008 1st International Conference on Information Technology (Gdansk, Poland, May 19 - 21, 2008). IT 2008. Gdansk University of Technology, Gdansk, Poland, 145-148.

[7] Scarfone, K. and Mell, P. Draft NIST Interagency Report 7502: The Common Configuration Scoring System (CCSS). NIST, May 2008. http://csrc.nist.gov/publications/PubsNISTIRs.html