The attached DRAFT document (provided here for HISTORICAL purposes) has been superseded by the following publication:

Publication Number:	**NIST Interagency Report 7517 (note document # change in final version – final version is NISTIR 7864)**
Title:	**The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities**
Publication Date:	**07/09/2012**

- Final Publication: http://dx.doi.org/10.6028/NIST.IR.7864
- Related Information on CSRC: http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7864
- Information on other NIST Computer Security Division publications and programs can be found at: http://csrc.nist.gov/

The following information was posted with the attached DRAFT document:

NIST Interagency Report (IR) 7864, The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities, has been released as final.
July 9, 2012

NIST Interagency Report (IR) 7864, The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities, has been released as final. This report proposes a specification for CMSS, a set of standardized measures for the severity of software feature misuse vulnerabilities. Software feature misuse vulnerabilities are vulnerabilities in which software features also provide an avenue to compromise the security of a system. NISTIR 7864 also provides examples of how CMSS measures and scores would be determined. CMSS data can assist organizations in making security decisions based on standardized, quantitative vulnerability data.

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

NIST Interagency Report 7517
(Draft)

The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities (DRAFT)

Elizabeth Van Ruitenbeek
Karen Scarfone

NIST Interagency Report 7517 (Draft)

The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities

Elizabeth Van Ruitenbeek
Karen Scarfone

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

February 2009

U.S. Department of Commerce

Carlos M. Gutierrez, 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 7517 (Draft)
37 pages (Feb. 2009)

Acknowledgements

The authors, Elizabeth Van Ruitenbeek, of both the University of Illinois at Urbana-Champaign and the National Institute of Standards and Technology (NIST), and Karen Scarfone of NIST wish to thank their colleagues who reviewed drafts of this report and contributed to its technical content, particularly Peter Mell and Tim Grance of NIST.

Portions of this report are based on the official Common Vulnerability Scoring System (CVSS) standard[1] from the Forum for Incident Response and Security Teams (FIRST) CVSS Special Interest Group; NIST Interagency Report (IR) 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*[2]*;* and draft NIST IR 7502, *The Common Configuration Scoring System (CCSS)*[3].

Abstract

The Common Misuse Scoring System (CMSS) consists of a set of measures of the severity of software feature misuse vulnerabilities. A software feature misuse vulnerability is present when the trust assumptions made when designing software features can be abused in a way that violates security. Misuse vulnerabilities allow attackers to use for malicious purposes the functionality that was intended to be beneficial. CMSS is derived from the Common Vulnerability Scoring System (CVSS), which was developed to score the severity of vulnerabilities due to software flaws. The CMSS measures are divided into three categories: base, temporal, and environmental. Base metrics assess the intrinsic exploitability of the vulnerability and the impact on confidentiality, integrity, and availability. Temporal metrics measure the time-varying aspects of vulnerability severity, such as the prevalence of exploits. Environmental metrics measure the aspects of vulnerability severity specific to an organization's environment, such as the local implementation of remediation measures. CMSS also includes a formula that combines those measures to produce a severity score for each vulnerability. CMSS enables organizations to make security decisions based on a standardized quantitative assessment of their vulnerability to software feature misuse.

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

List of Figures

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. 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.[4] A *security configuration issue vulnerability* involves the use of security configuration settings that negatively affect the security of the software.

A *software feature misuse vulnerability* is caused by the software designer making trust assumptions that permit the software to provide a beneficial feature, 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 ARP replies contain 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 incorrect 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] This text was derived from draft NIST IR 7502, *The Common Configuration Scoring System (CCSS)* (http://csrc.nist.gov/publications/PubsNISTIRs.html).

[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, software flaws may be caused by design errors, which also cause misuse vulnerabilities. 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—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 anti-phishing 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 can ever be 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 single password, and it would also permit attackers to perform denial-of-service attacks against users more easily by attempting to log in once to 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 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 administrator-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 an attacker 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 should 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 include the following. 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, it is often 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.

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 the past few years in using it much more broadly, such as inputs to risk assessment methodologies.

The second vulnerability measurement and scoring specification is the Common Configuration Scoring System (CCSS). Derived from CVSS, CCSS was designed for measuring and scoring software configuration issue vulnerabilities. CCSS uses the basic components of CVSS and adjusts them to account for the differences between software flaws and security configuration issues. A draft of CCSS was released in 2008 and is undergoing revisions as of this writing.[7]

The Common Misuse Scoring System (CMSS), the third of the vulnerability measurement and scoring specifications, is defined in this report. CMSS addresses software feature misuse vulnerabilities. CMSS is largely based on CVSS and CCSS, and it is intended to complement them.

The three vulnerability measurement and scoring systems are quite similar.[8] 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

---

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 7502, *The Common Configuration Scoring System (CCSS)*

8 There are some significant differences among the systems; these are discussed in Section 4 of this report.

significant differences in the three specifications. These differences are discussed in Section 4, after the CMSS specification has been defined and discussed in detail in Sections 2 and 3.

At a conceptual level, the CMSS specification can be more challenging to understand than the CVSS and CCSS specifications because of the open-endedness of misuse vulnerabilities. The vulnerabilities addressed by CVSS and CCSS are concrete: known software flaws and security configuration settings. They are defined in vulnerability dictionaries.[9] However, as of this writing there is not yet a dictionary of software feature misuse vulnerabilities. Creating such a dictionary will require systematic identification of the types of the trust assumptions that, if violated, could permit the flow of malicious code into a system, the flow of confidential data out of the system, or other consequences such as denial of service conditions or destruction of data. Consider the analysis of instant messaging (IM) software that allows a user to send and receive text. The user may trustingly assume that when text appears to come from a friend, the text was sent by that friend and can be trusted. However, an attacker may violate that trust when, for example, he gains control of the friend's IM client and sends the user a message containing the URL of a malicious website. This is a misuse vulnerability: that an attacker can masquerade as the user's IM friend, exploit the user's trust, and lead the user to compromise the security of his computer.

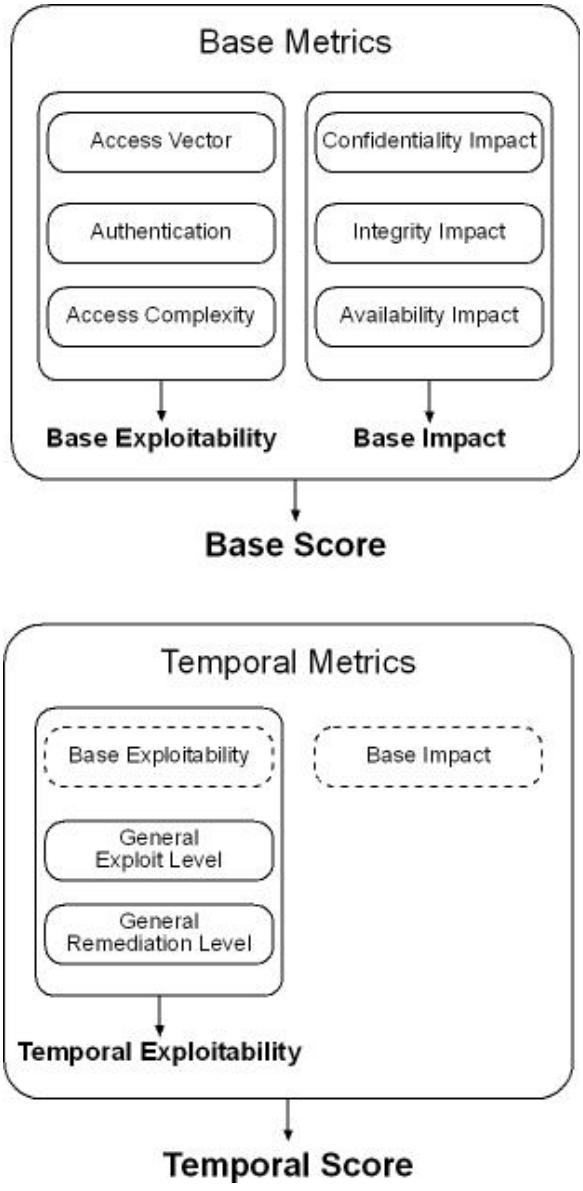
While some misuse vulnerability exploits begin with the attacker initiating contact, other exploits rely on the victim to seek them out. For example, a user may trust that the files downloaded from a peer-to-peer network are safe, but a misuse vulnerability exists if an attacker is able, for example, to misrepresent an infected file to users who naively download the file and infect their computers. Also, when a misuse vulnerability involves abusing the trust assumptions of people, an attack may include social engineering tactics that prey on aspects of human nature such as curiosity, greed, fear, or trust of authority. Social engineering can play an important role in exploiting misuse vulnerabilities; however, the discussion of specific social engineering techniques is beyond the scope of this report.

Section 4 contains additional information on a possible dictionary for software feature misuse vulnerabilities.

[9] The software flaw dictionary is Common Vulnerabilities and Exposures (CVE) (http://cve.mitre.org/), and the security configuration issue dictionary is Common Configuration Enumeration (CCE) (http://cce.mitre.org/).

2. CMSS Metrics

The CMSS metrics are organized into three groups: base metrics, temporal metrics, and environmental metrics. Base metrics describe the characteristics of a misuse vulnerability that are constant over time and across user environments. Temporal metrics describe the characteristics of misuse vulnerabilities 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.



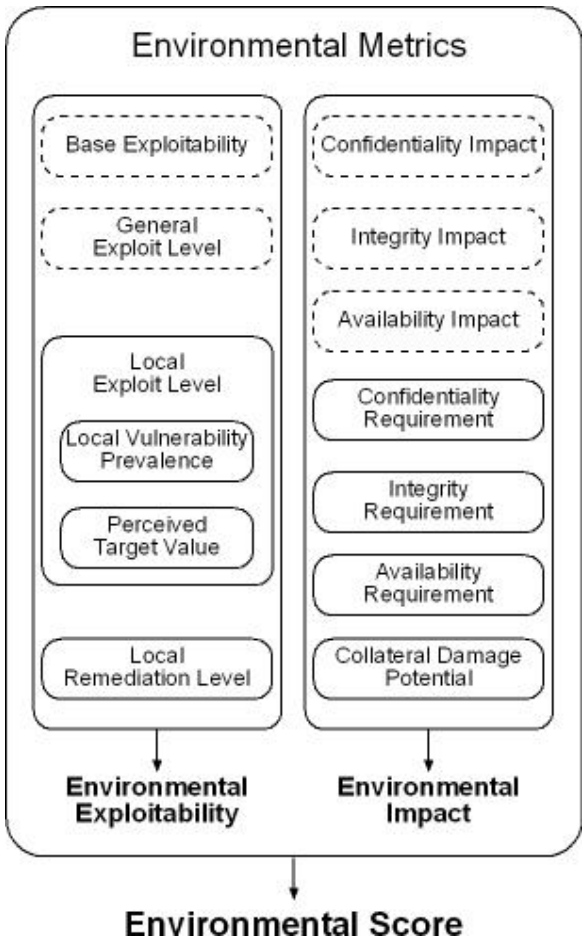


Figure 1. CMSS Metric Groups

2.1 Base Metrics

This section describes the base metrics, which measure the characteristics of a software feature misuse vulnerability 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 software feature misuse vulnerability can be captured using the Access Vector, Authentication, and Access Complexity metrics. These metrics are adapted from the CVSS specification and reinterpreted in the context of software feature misuse.

2.1.1.1 Access Vector (AV)

The Access Vector metric reflects the access required to exploit the vulnerability. To produce an Access Vector score for a software feature misuse vulnerability, consider what access to the system the attacker must possess in order to exploit this software feature. The possible values for this metric are listed in Table 1. The more remote an attacker can be to attack a host, the greater the vulnerability score.

Table 1. Access Vector Scoring Evaluation

Metric Value	Description
Local (L)	A vulnerability exploitable with only local access requires the attacker to have either physical access to the vulnerable system or a local (shell) account. An example of a locally exploitable misuse vulnerability is the use of synchronization software to transfer malicious code from a docked mobile device to the system.
Adjacent Network (A)	A vulnerability exploitable with adjacent network access requires the attacker 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 a misuse vulnerability exploitable using adjacent network access is a system with a Bluetooth interface that offers no security features (the interface can only be enabled or disabled). An attacker within range of the system's enabled Bluetooth interface could connect to the system through that interface and perform actions such as maliciously accessing and modifying files.
Network (N)	A vulnerability exploitable with network access means that the attacker does not require local network access or local access. An example of a network attack is the distribution of an infected email attachment that the recipients are tempted to open (which would be a misuse of the email file attachment feature).

2.1.1.2 Authentication (AU)

The Authentication metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability. This metric does not gauge the strength or complexity of the authentication process, only that an attacker is required to provide credentials before an exploit may occur. The possible values for this metric are listed in Table 3. The fewer authentication instances that are required, the higher the vulnerability 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 vulnerabilities, 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 vulnerability 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 vulnerability, then this metric should be set to "single."

Table 2. Authentication Scoring Evaluation

Metric Value	Description
Multiple (M)	Exploiting the vulnerability requires that the attacker authenticate two or more times, even if the same credentials are used each time. An example is an attacker 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 vulnerability.
None (N)	Authentication is not required to access and exploit the vulnerability.

The metric should be applied based on the authentication the attacker requires before launching an attack. For example, if a network service is vulnerable to a command that can be issued before a user

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

authenticates to the service, the metric should be scored as “None” because the attacker can launch the exploit before credentials are required. If the vulnerable command is only available after successful authentication, then the vulnerability should be scored as “Single” or “Multiple,” depending on how many instances of authentication must occur before issuing the command.

2.1.1.3 Access Complexity (AC)

The Access Complexity metric reflects the complexity of the attack required to exploit the software feature misuse vulnerability. When the misuse vulnerability is manifest by user action, the complexity of a software feature misuse attack depends both on the number of misuse actions the user must be persuaded to perform and the level of sophistication of the social engineering such persuasion requires. Otherwise, the complexity more generally depends on the level of sophistication required of the attacker to be able to exploit the misuse vulnerability. Access Complexity can be influenced by factors such as the ease of implementing and launching the attack and the likelihood of a user misusing the software feature in the manner desired by the attacker. Access Complexity increases when an attack depends on additional system requirements, such as using a particular type of web browser or a web browser with a particular type of active content enabled.

For example, first consider an enticing email containing malicious scripts that execute when the user views the email. The Access Complexity is medium because attack success requires a single user action that is relatively likely to occur. Other misuse vulnerabilities may require additional steps in order to be exploited. For example, an email may include a hyperlink to a website containing malicious code for the user to download and install. This indirect infection method would require the user to follow several steps to complete the exploit. To be successful, this attack would likely require sophisticated social engineering. Thus, the Access Complexity would be rated as high.

In contrast to the previous two examples, some vulnerability exploits require no direct user interaction, such as when an email client automatically displays emails (including rendering any malicious code they contain) without user consent and without the option to disable the feature. The Access Complexity of this vulnerability would be rated as low because the attacker can exploit the vulnerability essentially at will. Although the exploit requires that the victim run the email client, the user will presumably run the email client at some point in time.

The possible values for this metric are listed in Table 2. The lower the required complexity, the higher the vulnerability score.

Table 3. Access Complexity Scoring Evaluation

Metric Value	Description

High (H)	Specialized access conditions exist. For example:
	• For misuse vulnerabilities dependent on user actions, the misuse actions required of the user are unlikely to be performed.
	  o To enable the exploit, the user must perform complex or unusual steps, possibly within a sequence of steps (e.g., the user receives an instant message with a link to a website that contains a Trojan horse program that the user would have to select, download, and install).
	  o The attack depends on elaborate social engineering techniques that would be easily detected by knowledgeable people. For example, the user must be persuaded to perform suspicious or atypical actions.
	• The attacker must perform a complex sequence of steps to exploit the trust assumptions of programs running on the target host (e.g., the attacker must first compromise another program that the vulnerable program trusts).

Medium (M)	The access conditions are somewhat specialized. For example:
	• For misuse vulnerabilities dependent on user actions, the misuse actions required of the user are at least somewhat likely to be performed.
	  o The user must perform easy or seemingly ordinary steps to enable the exploit (e.g., the user runs the executable file attached to an email).
	  o The attack depends on a small amount of social engineering that might occasionally fool cautious users (e.g., phishing attacks that modify a web browser's status bar to show a false link, having to be on someone's "buddy" list before sending an IM exploit).
	• The attacker must perform moderately difficult steps to exploit the trust assumptions of programs running on the target host (e.g., the attacker must create an email message containing a malicious script).

Low (L)	Specialized access conditions or extenuating circumstances do not exist. For example: • The attack bypasses user consent mechanisms, if any exist; no user action is required. • The attacker must perform simple steps to exploit the trust assumptions of programs running on the target host (e.g., the attacker crafts a malicious Address Resolution Protocol (ARP) reply message to poison an ARP table with incorrect address mappings).

2.1.2 Impact

The Impact of a software feature misuse vulnerability 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 software feature misuse. These three Impact metrics measure how a misuse vulnerability, if exploited, will directly affect a targeted host. The Impact metrics reflect the degree of loss of confidentiality, integrity, and availability. For example, a vulnerability could cause a partial loss of integrity and availability, but no loss of confidentiality.

2.1.2.1 Confidentiality Impact (C)

The Confidentiality Impact metric measures the impact on confidentiality of a successfully exploited misuse vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. The possible values for this metric are listed in Table 4. Increased Confidentiality Impact increases the vulnerability 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 informational disclosure. Access to some system files is possible, but the attacker 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.
Complete (C)	There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system's data (memory, files, etc.)

2.1.2.2 Integrity Impact (I)

The Integrity Impact metric measures the impact to integrity of a successfully exploited misuse vulnerability. Integrity refers to the trustworthiness and guaranteed veracity of information. The possible values for this metric are listed in Table 5. Increased Integrity Impact increases the vulnerability score.

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 attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited. For example, system or application files may be overwritten or modified, but either the attacker has no control over which files are affected or the attacker can modify files within only a limited context or scope.
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 attacker is able to modify any files on the target system.

2.1.2.3 Availability Impact (A)

The Availability Impact metric measures the impact to availability of an exploited misuse vulnerability. 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. Increased Availability Impact increases the vulnerability 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. An example is a network-based flood attack that permits a limited number of successful connections to an Internet service.
Complete (C)	There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.

2.2 Temporal Metrics

The threat posed by a misuse vulnerability may change over time. The base metrics are limited to the characteristics of a software feature misuse vulnerability that are constant over time and across user environments. To incorporate the time-variant aspect of misuse vulnerability 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 misuse 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 the particular metric 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 misuse vulnerability. The prevalence of attacks determines how often any vulnerable system is likely to come under attack. If a misuse vulnerability 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 misuse vulnerability 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 prevalent the exploitation of a vulnerability, the higher the vulnerability 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 prevent misuse and mitigate the vulnerability. Remediation measures may restrict the usage of the feature to minimize or prevent misuse. Although misuse vulnerabilities can be removed by uninstalling the vulnerable software, the General Remediation Level only measures the availability of remediation

techniques that do not involve removal of the software. One example of a remediation measure available against users opening infected email attachments is the anti-virus check in an email client that restricts which attachments a user is able to open. Similarly, an anti-spam or anti-phishing filter in an email client can mitigate the effects of a phishing email by restricting which incoming email messages are placed in the inbox for the user to view and by alerting the user about suspected phishing sites. These measures restrict the usage of the email client in an attempt to prevent misuse of the capabilities to view emails and open attachments. 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 vulnerability score is.

Table 8. General Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are available to significantly restrict feature use in such a way to decrease the incident of misuse by between 76% and 100%. (An incident decrease of 100% means that the available remediation measures are able to entirely prevent misuse.)
Medium (M)	Remediation measures are available to partially restrict feature use in such a way to decrease the incident of misuse by between 26% and 75%.
Low (L)	Remediation measures are available to slightly restrict feature use in such a way to decrease the incident of misuse 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 an IT environment. Each organization computing CMSS metrics can determine an appropriate definition of IT environment. 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, the Local Exploit Level and the Local Remediation Level environmental metrics produce a scaling factor that is applied to the Exploitability components of the base metric. Local Impact environmental metrics produce both an additional impact component (Collateral Damage Potential) and scaling factors that are applied to the impact components of the base metric.

The environmental metrics are intended to measure deviations from the “typical” environment assumptions that were used to compute the base and temporal metrics. Therefore, environmental metrics should be scored relative to those “typical” 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 an environment. It is an environment-specific indicator 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 misused feature under scrutiny and the prevalence of its misuse. For misuse vulnerabilities dependent on user actions, the prevalence of misuse depends on the probability that users in this environment will perform the misuse actions required for vulnerability exploitation. For example, if 80% of the systems contain a particular potentially misused feature but only half of the user population of those systems are expected to engage in misuse behavior, 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 proportion of vulnerable systems, the higher the vulnerability 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 considerable 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 misuse vulnerability 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 misuse 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 vulnerability 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 misuse 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.

Remediation measures may restrict the usage of a feature to minimize or prevent misuse and thereby mitigate or remove the vulnerability. For example, to mitigate the misuse vulnerability present in web browsers displaying mobile code content (e.g., ActiveX or JavaScript), firewall rules may be used to block all such content (high local remediation), block content originating from sites not known to be secure (medium local remediation), block content known to be insecure (low local remediation), or block no mobile code content (local remediation level of none).

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

Table 11. Local Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are implemented to restrict feature use in such a way to decrease the incident of misuse by between 76% and 100%. (An incident decrease of 100% means that the implemented remediation measures entirely prevent misuse.)
Medium (M)	Remediation measures are implemented to partially restrict feature use in such a way to decrease the incident of misuse by between 26% and 75%.
Low (L)	Remediation measures are implemented to slightly restrict feature use in such a way to decrease the incident of misuse 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 four environmental metrics: Collateral Damage Potential, Confidentiality Requirement, Integrity Requirement, and Availability Requirement. Collateral Damage Potential is an additional impact metric that augments the three base impact metrics (Confidentiality Impact, Integrity Impact, and Availability Impact). The remaining three environmental metrics (Confidentiality Requirement, Integrity Requirement, and Availability Requirement) are used to compute scaling factors that are applied to the three base impact metrics.

2.3.3.1 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 host that uses the application in a trivial way versus another host 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. Clearly, each organization must determine for itself the precise meaning of "slight, moderate, significant, and catastrophic."

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)	A successful exploit of this vulnerability may result in slight physical or property damage. Or, there may be a slight loss of revenue or productivity to the organization.
Low-Medium (LM)	A successful exploit of this vulnerability may result in moderate physical or property damage. Or, there may be a moderate loss of revenue or productivity to the organization.
Medium-High (MH)	A successful exploit 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)	A successful exploit of this vulnerability may result in catastrophic physical or property damage and 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.

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

The Confidentiality Requirement, Integrity Requirement, and Availability Requirement metrics enable the analyst to customize the CMSS 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."

The full effect on the environmental score is determined by the corresponding base impact metrics. That is, these metrics modify the environmental score by reweighting the (base) Confidentiality, Integrity, and Availability Impact metrics.[11] For example, the Confidentiality Impact (C) metric has *increased* weight if the Confidentiality Requirement (CR) is "high." Likewise, the Confidentiality Impact metric has *decreased* weight if the Confidentiality Requirement is "low." The Confidentiality Impact metric weighting is neutral if the Confidentiality Requirement is "medium." This same logic is applied to the Integrity and Availability Requirements.

Note that the Confidentiality Requirement will not affect the environmental score if the (base) Confidentiality Impact is set to "none." Also, increasing the Confidentiality Requirement from "medium" to "high" will not change the environmental score when the (base) impact metrics are set to "complete." This is because the Impact subscore (the part of the base 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 vulnerability score. Remember that "medium" is considered the default.

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

[11] Please note that the base Confidentiality, Integrity and Availability Impact metrics, themselves, are not changed.

"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] CMSS follows this general model of FIPS 199, but does not require organizations to use any particular system 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

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]
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]/ 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" would have the following base vector: "AV:L/AC:M/Au:N/C:N/I:P/A:C." 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, 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/CDP:ND/CR:M/IR:H/AR:L."

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

3. Scoring

This section explains how CMSS scoring is performed. It first provides guidelines on performing scoring. Next, it defines the equations used for base, temporal, and environmental score generation. Finally, it 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 vulnerabilities. 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.

3.1.1 General

SCORING TIP #1: Vulnerability scoring should not take into account any interaction with other vulnerabilities. That is, each vulnerability should be scored independently.

SCORING TIP #2: When scoring the base metrics for a vulnerability, consider the direct impact to the target host only.

SCORING TIP #3: Many applications, such as Web servers, can be run with different privileges, and scoring the impact involves making an assumption as to what privileges are used. Therefore, vulnerabilities should be scored according to the generally accepted best practice for the privileges. This may not necessarily reflect common practices, especially for client applications which are often run with root-level privileges.[13] When uncertain as to which privileges are considered the best practice, scoring analysts should assume a default configuration.

SCORING TIP #4: When scoring the impact of a vulnerability that has multiple exploitation methods (attack vectors), the analyst should compute and report multiple scores.

3.1.2 Base Metrics

3.1.2.1 Access Vector

SCORING TIP #5: When a vulnerability can be exploited both locally and from the network, the "Network" value should be chosen. When a vulnerability can be exploited both locally and from adjacent networks, but not from remote networks, the "Adjacent Network" value should be chosen. When a vulnerability can be exploited from the adjacent network and remote networks, the "Network" value should be chosen.

SCORING TIP #6: Many client applications and utilities have local vulnerabilities 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. Also, helper applications (office

[13] An option for addressing this is to add a base metric that measures the level of access that can be gained by exploiting the vulnerability, such as gaining access with the privileges of the user. Organizations with implementations that permit more privileges than the generally accepted best practice could then alter the values of the impact metrics for their temporal and environmental calculations. For example, gaining user-level access when the user account has limited privileges might have base impact metrics of Partial/Partial/Partial; if the user account has administrator-level privileges, then this could be changed to impact metrics of Complete/Complete/Complete. Readers are particularly encouraged to provide feedback on this CMSS design decision.

suites, image viewers, media players, etc.) are exploited when malicious files are exchanged via e-mail or downloaded from web sites. Therefore, analysts should score the Access Vector of these vulnerabilities as "Network".

3.1.2.2 Authentication

SCORING TIP #7: If the vulnerability exists in an authentication scheme itself (e.g., Pluggable Authentication Module [PAM], Kerberos) or an anonymous service (e.g., public FTP server), the metric should be scored as "None" because the attacker can exploit the vulnerability without supplying valid credentials. Presence of a default user account may be considered as "Single" or "Multiple" Authentication (as appropriate), but may have Exploitability of "High" if the credentials are publicized.

3.1.2.3 Confidentiality, Integrity, Availability Impacts

SCORING TIP #8: Vulnerabilities that give root-level access should be scored with complete loss of confidentiality, integrity, and availability, while vulnerabilities that give user-level access should be scored with only partial loss of confidentiality, integrity, and availability. For example, an integrity violation that allows an attacker to modify an operating system password file should be scored with complete impact of confidentiality, integrity, and availability.

SCORING TIP #9: Vulnerabilities with a partial or complete loss of integrity can also cause an impact to availability. For example, an attacker who is able to modify records can probably also delete them.

3.2 Equations

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

3.2.1 Base Equation

The base equation is the foundation of CMSS scoring. The 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
```

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. The temporal GeneralRemediationLevel metric 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 equation:

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

EnvironmentalImpact = min(10, 10.41*(1-(1-ConfImpact*ConfReq)*
      (1-IntegImpact*IntegReq)*(1-AvailImpact*AvailReq))
      *(CollateralDamagePotential))

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

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
```

3.3 Examples

The examples below show how CMSS is used to score software feature misuse vulnerabilities.

3.3.1 Example One: ARP Cache Poisoning

The Address Resolution Protocol (ARP) trusts that ARP replies contain 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, a misuse vulnerability exists when an attacker can poison the ARP table with incorrect address mappings and thereby launch a denial-of-service or a man-in-the-middle attack.

Since the attacker must have access to the local subnetwork to send malicious ARP replies, the Access Vector is "Adjacent Network." No authentication is required to broadcast ARP replies, so the Authentication is scored as "None." The Access Complexity is "Low" because exploitation of the vulnerability requires little skill on the part of the attacker. The attacker must craft a message in valid ARP reply format; the ARP reply message may contain arbitrary IP and MAC addresses.

The impact metrics measure only the *direct* impact of exploitation of the vulnerability. The Confidentiality Impact of this misuse vulnerability is "None" because there is no direct impact on the confidentiality of the system. The Integrity Impact is "Partial" because the attacker can override valid ARP cache entries and can add false entries. The attacker can only modify data in this limited context. The Availability Impact is "Partial" because ARP cache poisoning can create a denial of service that impacts the availability of network functions, yet non-network functions remain available.

With the base metric scores described above, the base score for this misuse vulnerability is 4.8.

Temporal metrics describe the general prevalence of attacks against this vulnerability and the general availability of remediation measures. The General Remediation Level for the ARP cache poisoning vulnerability would be considered "Low" because there are limited mitigation techniques available. For very small networks, administrators can configure static IP addresses and static ARP tables, but this approach quickly becomes unmanageable as the network grows in size. For larger networks, switches can be configured to allow only one MAC address for each physical port. ARP cache poisoning attacks occur against typical systems rarely, so the General Exploit Level is scored as "Low". Since the General Remediation Level is also scored as "Low," the temporal score would be 3.7. In general, the temporal score can be lower than the base score when the General Exploit Level is lower than "Medium" or the General Remediation Level is higher than "None."

Environmental metrics describe the vulnerability severity with respect to a particular organization. Consider an organization in which the Local Vulnerability Prevalence is "High," the Perceived Target Value is "Medium", and the Local Remediation Level is rated "None." Because the Local Vulnerability Prevalence is higher than the default value and the Local Remediation Level is lower than the General Remediation Level, the exploitability subscore calculation for the environmental score is higher than the temporal exploitability subscore.

Now consider the impact subscore of the environmental score. When the Collateral Damage Potential is "None," this metric does not modify the impact subscore in the environmental score calculation. Scores of "Medium" for Confidentiality Requirement and Availability Requirement also do not modify the impact subscore. However, since this vulnerability impacts integrity (recall that the base Integrity Impact is "Partial"), a "High" score for Integrity Requirement increases the impact subscore. The final computation

combines the environmental subscores for exploitability and impact and produces an environmental score of 5.4.

```
-----------------------------------------------------
BASE METRIC                 EVALUATION         SCORE
-----------------------------------------------------
Access Vector               [Adjacent]        (0.646)
Authentication              [None]            (0.704)
Access Complexity           [Low]             (0.71)
Confidentiality Impact      [None]            (0.0)
Integrity Impact            [Partial]         (0.275)
Availability Impact         [Partial]         (0.275)
-----------------------------------------------------
BASE FORMULA                                BASE SCORE
-----------------------------------------------------
Impact = 10.41*(1-(1)*(0.725)*(0.725)) == 4.94
Exploitability = 20*0.646*0.704*0.71 == 6.46
f(Impact) = 1.176
BaseScore = (0.6*4.94 + 0.4*6.46 - 1.5)*1.176
                                              == (4.8)
-----------------------------------------------------

-----------------------------------------------------
TEMPORAL METRIC             EVALUATION         SCORE
-----------------------------------------------------
General Exploit Level       [Low]              (0.8)
General Remediation Level   [Low]              (0.8)
-----------------------------------------------------
TEMPORAL FORMULA                         TEMPORAL SCORE
-----------------------------------------------------
TemporalExploitability = min(10,6.46*0.8*0.8) == 4.13
TemporalScore = (0.6*4.94 + 0.4*4.13 – 1.5)*1.176
                                              == (3.7)
-----------------------------------------------------

-----------------------------------------------------
ENVIRONMENTAL METRIC        EVALUATION         SCORE
-----------------------------------------------------
Local Vulnerability Prevalence [High]          (1.2)
Perceived Target Value         [Medium]        (1.0)
Local Remediation Level        [None]          (1.0)
Collateral Damage Potential    [None]          (1.0)
Confidentiality Req.           [Medium]        (1.0)
Integrity Req.                 [High]          (1.51)
Availability Req.              [Medium]        (1.0)
-----------------------------------------------------
ENVIRONMENTAL FORMULA            ENVIRONMENTAL SCORE
-----------------------------------------------------
LocalExploitLevel = 1.2*1.0 == 1.2
EnvironmentalExploitability =
   min(10,6.46*0.8*1.2*1.0) == 6.20
EnvironmentalImpact =
   min(10, 10.41*(1-((1)*(1-0.275*1.51)*(1-0.275*1.0)))
       *1.0) == 6.00
EnvironmentalScore =
   (0.6*6.00 + 0.4*6.20 - 1.5)*1.176
                                              == (5.4)
-----------------------------------------------------
```

3.3.2 Example Two: Malicious File Transfer Via Instant Messaging Software

Instant messaging (IM) software allows a user to send and receive files. The user may trustingly assume that when a file appears to come from a friend, the file was sent by that friend and can be trusted. However, an attacker may violate that trust when he sends a malicious file that appears to come from the friend. (This could be accomplished in several ways, such as the attacker gaining control of the friend's IM client, the attacker spoofing the friend's IM user identity, or the attacker using social engineering to trick the friend into sending the file. The method used to accomplish this is irrelevant in terms of the user's vulnerability.) This is a misuse vulnerability: that an attacker can masquerade as the user's IM friend, exploit the user's trust, and lead the user to compromise the security of his computer.

Since an attacker can exploit this vulnerability remotely, the Access Vector is "Network." The Authentication is scored as "None" because the attacker does not need to authenticate to the target computer. To enable the exploitation of this vulnerability, the user must perform an easy, ordinary action (accepting and downloading a file appearing to come from a friend). The success of this attack depends on social engineering that could occasionally fool cautious users. Thus, the Access Complexity is rated "Medium."

The direct impact of this vulnerability affects the integrity of the target computer. By exploiting this vulnerability, the attacker can place a malicious file on the user's computer. Placing untrusted code on the target computer results in a "Partial" impact on the computer's integrity. There is no impact on confidentiality because the attacker is not accessing any information or resources from the computer. There is also no impact on availability because the transfer of untrusted code onto a machine does not directly impact availability[14].

With the base metric scores described above, the base score for this misuse vulnerability is 4.3.

Temporal metrics amend the base score to incorporate the general prevalence of attacks against this vulnerability and the general availability of remediation measures. Since attacks against this IM file transfer vulnerability are relatively infrequent, the General Exploit Level would be rated as "Low." The General Remediation Level would be "None" because there are no remediation measures available besides uninstalling the vulnerable IM software. Thus, the temporal score would be 3.5.

Environmental metrics describe the vulnerability severity with respect to a particular organization. Consider an organization in which the Local Vulnerability Prevalence is "Medium," the Perceived Target Value is "Low", and the Local Remediation Level is rated "None." Because the Perceived Target Value is less than the default value of "Medium" (and the other score components are at the default values), the exploitability subscore calculation for the environmental score is lower than the temporal exploitability subscore.

The environmental score also includes an impact subscore. The Confidentiality Requirement and Integrity Requirement are scored as "Medium," which does not modify the impact subscore. The Availability Requirement is rated "Low," but this value has no effect on the impact subscore because the IM file transfer vulnerability has no impact on availability (recall that the base Availability Impact is "None"). In general, when some component of the base impact (Confidentiality Impact, Integrity Impact, or Availability Impact) is scored as "None," the respective security requirement component (Confidentiality Requirement, Integrity Requirement, or Availability Requirement) has no effect on the environmental

[14] Executing the untrusted code could overwrite a system or application file and make a service or application unavailable on the user's computer, but this is an indirect impact of the IM file transfer misuse vulnerability, not a direct impact, so it is not included in the metrics for this vulnerability.

score calculation. The final environmental impact component Collateral Damage Potential is “None” and does not modify the base impact subscore. Thus, the environmental score is 2.8.

```
-----------------------------------------------------
BASE METRIC                 EVALUATION         SCORE
-----------------------------------------------------
Access Vector               [Network]         (1.0)
Authentication              [None]            (0.704)
Access Complexity           [Medium]          (0.61)
Confidentiality Impact      [None]            (0.0)
Integrity Impact            [Partial]         (0.275)
Availability Impact         [None]            (0.0)
-----------------------------------------------------
BASE FORMULA                                BASE SCORE
-----------------------------------------------------
Impact = 10.41*(1-(1)*(0.725)*(1)) == 2.86
Exploitability = 20*1.0*0.704*0.61 == 8.59
f(Impact) = 1.176
BaseScore = (0.6*2.86 + 0.4*8.59 - 1.5)*1.176
                                              == (4.3)
-----------------------------------------------------

-----------------------------------------------------
TEMPORAL METRIC             EVALUATION         SCORE
-----------------------------------------------------
General Exploit Level       [Low]              (0.8)
General Remediation Level   [None]             (1.0)
-----------------------------------------------------
TEMPORAL FORMULA                         TEMPORAL SCORE
-----------------------------------------------------
TemporalExploitability = min(10,8.59*0.8*1.0) == 6.87
TemporalScore = (0.6*2.86 + 0.4*6.87 – 1.5)*1.176
                                              == (3.5)
-----------------------------------------------------

-----------------------------------------------------
ENVIRONMENTAL METRIC        EVALUATION         SCORE
-----------------------------------------------------
Local Vulnerability Prevalence [Medium]        (1.0)
Perceived Target Value         [Low]           (0.8)
Local Remediation Level        [None]          (1.0)
Collateral Damage Potential    [None]          (1.0)
Confidentiality Req.           [Medium]        (1.0)
Integrity Req.                 [Medium]        (1.0)
Availability Req.              [Low]           (0.5)
-----------------------------------------------------
ENVIRONMENTAL FORMULA            ENVIRONMENTAL SCORE
-----------------------------------------------------
LocalExploitLevel = 1.0*0.8 == 0.8
EnvironmentalExploitability =
   min(10,8.59*0.8*0.8*1.0) == 5.50
EnvironmentalImpact =
   min(10, 10.41*(1-((1)*(1-0.275*1.0)*(1)))
       *1.0) == 2.86
EnvironmentalScore =
   (0.6*2.86 + 0.4*5.50 - 1.5)*1.176
                                              == (2.8)
-----------------------------------------------------
```

3.3.3 Example Three: User Follows Link to Spoofed Web Site

Emails, instant messages, and other forms of electronic communication frequently contain hyperlinks to Web sites. An attacker may distribute a malicious hyperlink that surreptitiously leads a user to a spoofed Web site. When the user clicks on the malicious link, the Web browser displays a look-alike imitation of a legitimate site (often a banking or e-commerce site). The vulnerability is that a hyperlink purporting to lead to a legitimate site instead takes the user to a malicious site. The hyperlink capability is misused.

The Access Vector for this misuse vulnerability is "Network" because the attacker providing the link and operating the phishing site does not require local network access or local access to the user's computer. The Authentication is "None" because the attacker is not required to authenticate to exploit this vulnerability. To enable the exploitation of this vulnerability, the user must perform an easy, ordinary step (clicking on a hyperlink). The attack depends on social engineering that could occasionally fool cautious users (when the link and the site look okay to the casual observer). Therefore, the Access Complexity is "Medium."

The impact subscore for this misuse vulnerability considers only the direct impact of a hyperlink exploit. The direct Confidentiality Impact is "None." Even though users may subsequently choose to enter personal information at a phishing site, this loss of confidentiality is only an indirect impact from clicking on a hyperlink to a spoofed site. The Integrity Impact is "Partial" because the link to the spoofed website is not trustworthy. From the viewpoint of the user, the integrity of the hyperlink is compromised because the link does not lead to the Web site to which it appears to lead. The Availability Impact is "None" because the existence of a malicious hyperlink to a spoofed site does not prevent access to the legitimate site using the correct URL. Thus, the base score for this misuse vulnerability is 4.3.

Temporal metrics describe the prevalence of attacks against a misuse vulnerability and the availability of remediation measures. The General Exploit Level would be "Medium" because exploits of this nature are frequently observed. The General Remediation Level would be "Medium" because several technical measures exist that can alert users about suspected spoofed Web sites or block emails containing links to known phishing sites. Some Web browsers include anti-phishing toolbars or maintain blacklists of known phishing sites. With these additional factors taken into consideration, the temporal score is 2.7.

Environmental metrics describe the vulnerability severity with respect to a particular organization. Consider an organization in which the Local Vulnerability Prevalence is "High," the Perceived Target Value is "High", and the Local Remediation Level is rated "Medium." Because the Local Vulnerability Prevalence and the Perceived Target Value are higher than the default value of "Medium" (and the Local Remediation Level is the same as the General Remediation Level), the exploitability subscore calculation for the environmental score is higher than the temporal exploitability subscore.

The environmental score also includes an impact subscore. Consider an organization in which the Collateral Damage Potential is "Low" (higher than the default value "None"), the Confidentiality Requirement and Integrity Requirement are "High", and the Availability Requirement is "Medium." Since this misuse vulnerability has a "Partial" score for Integrity Impact, the "High" Integrity Requirement will boost the severity rating of the vulnerability in the portion of the score related to integrity impact. For this vulnerability, the Collateral Damage Potential component will also increase the severity rating in the impact subscore. The overall environmental score is 5.5.

Note that the misuse vulnerabilities in examples two and three receive the same base score; however, differences in the temporal metric components and environmental metric components produce different temporal and environmental scores for the two vulnerabilities.

```
-----------------------------------------------------
BASE METRIC                  EVALUATION         SCORE
-----------------------------------------------------
Access Vector                [Network]         (1.0)
Authentication               [None]            (0.704)
Access Complexity            [Medium]          (0.61)
Confidentiality Impact       [None]            (0.0)
Integrity Impact             [Partial]         (0.275)
Availability Impact          [None]            (0.0)
-----------------------------------------------------
BASE FORMULA                                BASE SCORE
-----------------------------------------------------
Impact = 10.41*(1-(1)*(0.725)*(1)) == 2.86
Exploitability = 20*1.0*0.704*0.61 == 8.59
f(Impact) = 1.176
BaseScore = (0.6*2.86 + 0.4*8.59 - 1.5)*1.176
                                              == (4.3)
-----------------------------------------------------

-----------------------------------------------------
TEMPORAL METRIC              EVALUATION         SCORE
-----------------------------------------------------
General Exploit Level        [Medium]           (1.0)
General Remediation Level    [Medium]           (0.6)
-----------------------------------------------------
TEMPORAL FORMULA                         TEMPORAL SCORE
-----------------------------------------------------
TemporalExploitability = min(10,8.59*1.0*0.6) == 5.15
TemporalScore = (0.6*2.86 + 0.4*5.15 – 1.5)*1.176
                                              == (2.7)
-----------------------------------------------------

-----------------------------------------------------
ENVIRONMENTAL METRIC         EVALUATION         SCORE
-----------------------------------------------------
Local Vulnerability Prevalence [High]           (1.2)
Perceived Target Value       [High]             (1.2)
Local Remediation Level      [Medium]           (0.6)
Collateral Damage Potential  [Low]              (1.25)
Confidentiality Req.         [High]             (1.51)
Integrity Req.               [High]             (1.51)
Availability Req.            [Medium]           (1.0)
-----------------------------------------------------
ENVIRONMENTAL FORMULA            ENVIRONMENTAL SCORE
-----------------------------------------------------
LocalExploitLevel = 1.2*1.2 == 1.44
EnvironmentalExploitability =
   min(10,8.59*1.0*1.44*0.6) == 7.42
EnvironmentalImpact =
   min(10, 10.41*(1-((1)*(1-0.275*1.51)*(1)))
       *1.25) == 5.40
EnvironmentalScore =
   (0.6*5.40 + 0.4*7.42 - 1.5)*1.176
                                              == (5.5)
-----------------------------------------------------
```

4. Comparing CMSS to CVSS and CCSS

CMSS is based on CVSS and CCSS, 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 invited to compare the specifications side-by-side. The specifications are all structured similarly, making such comparisons easy.[15]

For the base metrics, all three specifications use the same six metrics and the same formulas 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 in the draft specification for CCSS, which defines two types of exploitation: active and 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 original draft specification for CCSS does not define any temporal or environmental metrics, although these will be developed in the future. The temporal and environmental components of CMSS are based on those from CVSS, but have major differences. The temporal metrics in CVSS measured 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 misuse vulnerabilities addressed by CMSS, because misuse vulnerabilities can be used without exploit code, do not have complete remediations such as patches, and are already known to exist. So CMSS has a different set of temporal metrics 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.

CMSS offers a more complex set of environmental metrics than CVSS. CVSS has three: Collateral Damage Potential, Target Distribution, and Security Requirements. These metrics are all part of CMSS as well, although Target Distribution has been renamed Local Vulnerability Prevalence. Two other metrics have been added to CMSS: Perceived Target Value, which measures how attackers value the targets in the environment as opposed to other environments, and Local Remediation Level, which measures the effectiveness of mitigation measures in the local environment. CMSS also divides its environmental metrics into two groups: exploit and impact. This allows exploitation and impact environmental subscores to be generated for CMSS; such subscores are not available in CVSS.

[15] The other specifications are NIST IR 7435 and draft NIST IR 7502 (http://csrc.nist.gov/publications/PubsNISTIRs.html).

5. Conclusions and Future Work

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

A software feature is a functional capability provided by software. Software feature misuse is a vulnerability in which the feature also provides an avenue to compromise the security of a system. By using CMSS, organizations can measure and manage their security risk due to misuse vulnerabilities. The scores produced from CMSS enable informed security decisions that consider the trade-off between functionality and exposure to misuse vulnerabilities.

As a standardized scoring system, CMSS allows analysts to track the security of software through multiple versions and to quantitatively determine whether the overall security related to misuse vulnerabilities is improving or declining over time. Software vendors can also use the standardized scoring system to communicate to their customers the severity of the vulnerabilities in their products.

There is still significant work to be done before CMSS is ready for organizations to adopt. The most important missing element is a dictionary of misuse vulnerabilities. Once such a dictionary has been developed and the CMSS specification finalized, then measures and scores can be assigned to each entry and shared with the security community. This data can be used in conjunction with the CVSS and CCSS measures and scores 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.

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

6. Appendix A—Additional Resources

The following are resources related to CMSS and the related CVSS and CCSS specifications.

- The CVSS version 2 specification is available at http://www.first.org/cvss/cvss-guide.html. General information on CVSS's development is documented at http://www.first.org/cvss/.
- NISTIR 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*, describes the CVSS version 2 specification. The report is available at http://csrc.nist.gov/publications/PubsNISTIRs.html.
- Draft NISTIR 7502, *The Common Configuration Scoring System (CCSS)*, describes the CCSS specification. The draft report is available at http://csrc.nist.gov/publications/PubsNISTIRs.html.

7. Appendix B—Acronyms and Abbreviations

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

A Adjacent Network
A Availability
AC Access Complexity
AR Availability Requirement
ARP Address Resolution Protocol
AU Authentication
AV Access Vector

C Complete
C Confidentiality
CDP Collateral Damage Potential
CERT/CC CERT Coordination Center
CR Confidentiality Requirement
CCSS Common Configuration Scoring System
CMSS Common Misuse Scoring System
CVSS Common Vulnerability Scoring System

E Exploitability

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
IM Instant Messaging
IP Internet Protocol
IR Integrity Requirement
IT Information Technology
ITL Information Technology Laboratory

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

M Medium
M Multiple
MAC Media Access Control
MH Medium-High

N	Network
N	None
ND	Not Defined
NIAC	National Infrastructure Advisory Council
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
P	Partial
PAM	Pluggable Authentication Module
PTV	Perceived Target Value
RL	Remediation Level
S	Single
US-CERT	United States Computer Emergency Readiness Team