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

NIST Interagency Report 7864

http://dx.doi.org/10.6028/NIST.IR.7864

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

Elizabeth LeMay
Karen Scarfone
Peter Mell

NIST Interagency Report 7864

http://dx.doi.org/10.6028/NIST.IR.7864

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

Elizabeth LeMay
University of Illinois at Urbana-Champaign

Karen Scarfone
Scarfone Cybersecurity

Peter Mell
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD

COMPUTER SECURITY

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

July 2012

U.S. Department of Commerce

Rebecca M. Blank, Acting Secretary

National Institute of Standards and Technology

Patrick D. Gallagher, Under Secretary of Commerce for
Standards and Technology and 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 analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems.

Authority

This publication has been developed by NIST to further its statutory responsibilities under the Federal Information Security Management Act (FISMA), Public Law (P.L.) 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Interagency Report 7864
39 pages (July 2012)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by Federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, Federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST publications, other than the ones noted above, are available at http://csrc.nist.gov/publications.

Comments on this publication may be submitted to:

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930), Gaithersburg, MD 20899-8930

Acknowledgments

The authors, Dr. Elizabeth LeMay of the University of Illinois at Urbana-Champaign, Karen Scarfone of Scarfone Cybersecurity, and Peter Mell of the National Institute of Standards and Technology (NIST), wish to thank their colleagues who reviewed drafts of this report and contributed to its technical content, particularly Celia Paulsen, Harold Booth, and Tim Grance of NIST, Sasha Romanosky of Carnegie Mellon University, Steven Christey of the MITRE Corporation, Adam Shostack of Microsoft, and Anton Chuvakin.

Portions of this report are based on the official Common Vulnerability Scoring System (CVSS) standard[1] from the Forum of 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;* and NIST IR 7502, *The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities.*[2]

Abstract

The Common Misuse Scoring System (CMSS) is a set of measures of the severity of software feature misuse vulnerabilities. A software feature is a functional capability provided by software. A software feature misuse vulnerability is a vulnerability in which the feature also provides an avenue to compromise the security of a system. Such vulnerabilities are present when the trust assumptions made when designing software features can be abused in ways that violate security. Misuse vulnerabilities allow attackers to use for malicious purposes the functionality that was intended to be beneficial. CMSS can provide measurement data to assist organizations in making sound decisions on addressing software feature misuse vulnerabilities and in conducting quantitative assessments of the overall security posture of a system. This report defines proposed measures for CMSS and equations to be used to combine the measures into severity scores for each vulnerability. The report also provides examples of how CMSS measures and scores would be determined for selected software feature misuse vulnerabilities.

Keywords

Common Configuration Scoring System (CCSS); Common Misuse Scoring System (CMSS); Common Vulnerability Scoring System (CVSS); security metrics; vulnerabilities; vulnerability scoring

Audience

This report is directed primarily at information security researchers, particularly those interested in vulnerability measurement or security automation; security product vendors; and vulnerability analysts.

[1] http://www.first.org/cvss/cvss-guide.html

[2] 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 discusses existing 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.[3] These categories are described below.

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

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

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

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

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

ARP protocol was standardized over 25 years ago[4], 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.

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

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

1.2 The Need for Vulnerability Measurement and Scoring

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

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

If vulnerability characteristics related to these two concepts were measured and documented in a consistent, methodical way, the measurement data could be used by quantitative risk assessment methodologies for determining which vulnerabilities are most important for an organization to address using its limited resources. For example, when planning the security configuration settings for a new system, an organization could use vulnerability measurements as part of determining the relative

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

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.

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

Although having a set of measures for a vulnerability provides the level of detail necessary for in-depth analysis, sometimes it is more convenient for people to have a single measure for each vulnerability. So quantitative measures can be combined into a score—a single number that provides an estimate of the overall severity of a vulnerability. Vulnerability scores are not as quantitative as the measures that they are based on, so they are most helpful for relative 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 4.[5] Small scoring differences, such as vulnerabilities with scores of 4.8 and 5.1, do not necessarily indicate a significant difference in severity because of the margin of error in individual measures and the equations that combine those measures.[6]

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.[7] CVSS has been widely adopted by the Federal government, industry, and others. CVSS was originally intended for use in prioritizing the deployment of patches, but there has been considerable interest in applying it more broadly, such as using its measures as inputs to risk assessment methodologies.

The second vulnerability measurement and scoring specification is the Common Configuration Scoring System (CCSS).[8] CCSS was designed for measuring and scoring software security 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.

[5] CMSS is ordinal scoring, not cardinal. For example, a score of 10 isn't twice as bad as a score of 5.

[6] See http://www.first.org/cvss/history (current as of May 31, 2012) for more information on the margin of error and the origin of the equations. To summarize, scoring differences less than 0.5 are not intended to be statistically significant. The scores were arrived at heuristically with the intention of providing an even spread of scores across the possible range.

[7] 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).

[8] NIST IR 7502, *The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities*

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 similar. They all use the same six core measures to capture the fundamental characteristics of vulnerabilities. They all generate vulnerability severity scores in the range of 0 (lowest severity) to 10 (highest severity). However, there are also some significant differences in the details of the three specifications. These differences are discussed in Section 5, after the CMSS specification has been defined and illustrated in Sections 2, 3, and 4.

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.

The primary purpose of this document is to define CMSS, and not to explain in detail how organizations can use CMSS. Unlike CVSS data, which can be used by itself to aid in prioritizing vulnerability remediation efforts, as of this writing CMSS data is not yet directly useful to organizations for decision making. This document is an early step in a long-term effort to provide standardized vulnerability measurement data sources and corresponding methodologies for conducting quantitative risk assessments of system security. Additional information will be published in the future regarding CMSS and how organizations will be able to take advantage of it. Currently, the focus is on reaching consensus on the definition of CMSS and creating initial sets of CMSS measures and scores for the purposes of validating the CMSS specification and identifying any necessary adjustments to the specification.

Additional work will be needed 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, then measures and scores need to be assigned to each entry and shared with the security community. This data could be used in conjunction with the CVSS and CCSS measures as a consistent set of measures for

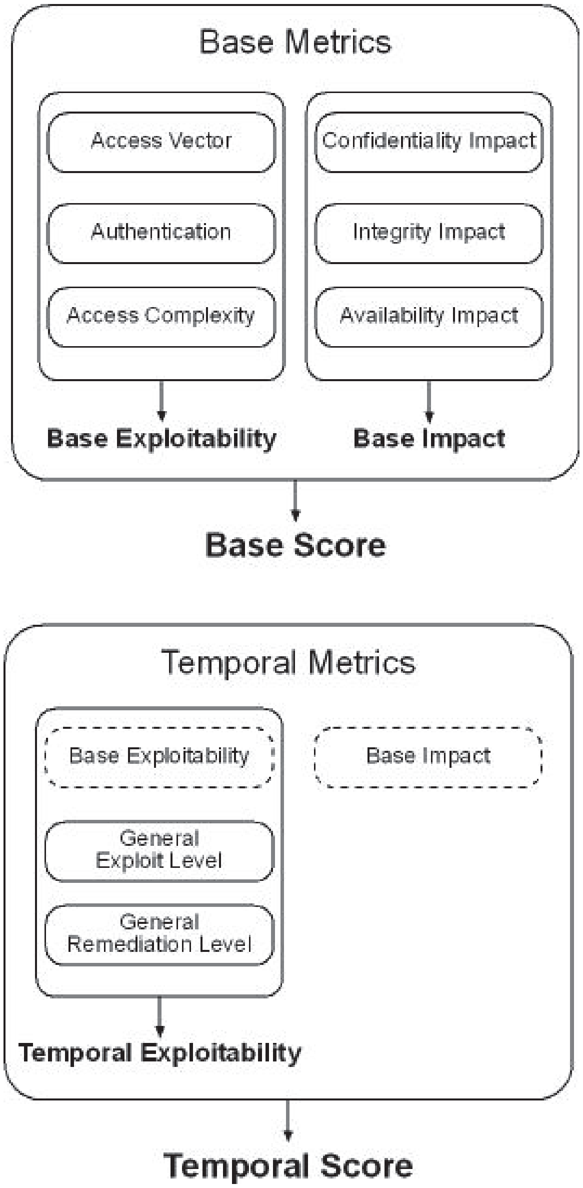
[9] The software flaw dictionary is Common Vulnerabilities and Exposures (CVE) (http://cve.mitre.org/), and the security configuration issue dictionary is Common Configuration Enumeration (CCE) (http://cce.mitre.org/).

system vulnerabilities. In turn, this will provide opportunities for using the data in threat models, risk assessments, and other security analysis activities.

Also, a way will need to be developed to relate data on various vulnerabilities to each other—there are many dependencies among vulnerabilities that affect their exploitability and impact. For example, one vulnerability might only be exploitable if a second vulnerability is also present or if a second vulnerability can grant user-level access. These dependencies need to be captured in a standardized way to facilitate the data's use for security modeling and analysis. Also, a mechanism has not yet been developed for measuring the security characteristics of a system using CVSS, CCSS, and CMSS together. Another important problem to resolve is how local policy, security controls, and other system and environment-specific elements that affect vulnerabilities should be taken into account.

2. CMSS Metrics

This section defines the metrics comprising the CMSS specification. The CMSS metrics are organized into three groups: base, temporal, and environmental. 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.



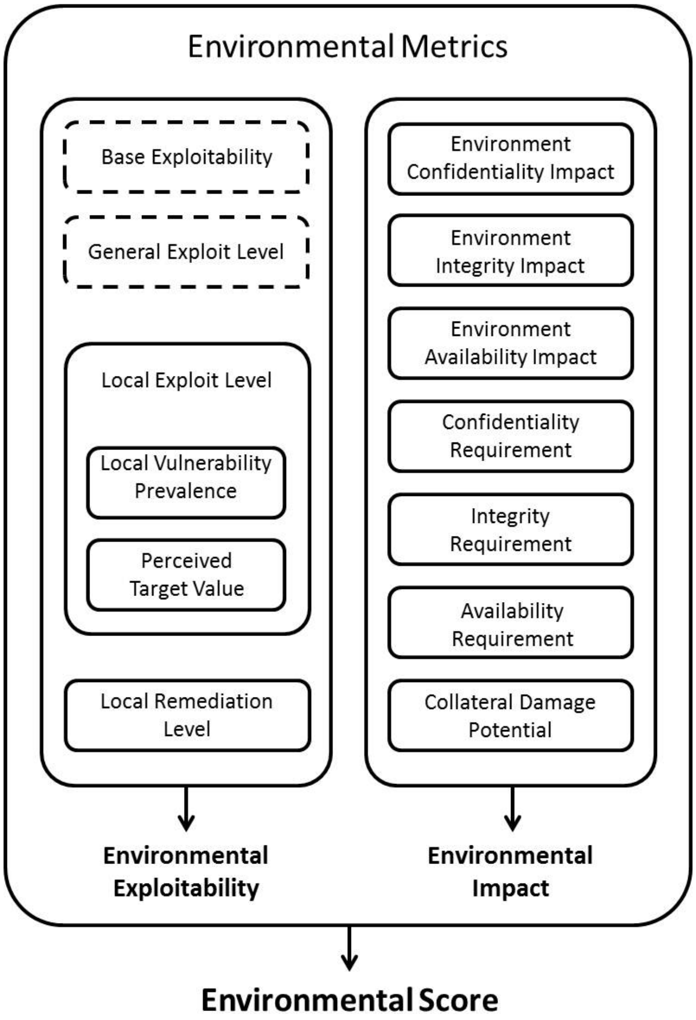


Figure 1. CMSS Metric Groups

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 and CCSS specifications 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 the feature. The possible values for this metric are listed in Table 1. The more remote an attacker can be, 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 email with an infected 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 2. 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.

[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.

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 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 3. 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 several suspicious or atypical actions. • The attacker must perform a complex, difficult sequence of steps to exploit the trust assumptions of programs running on the target system (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, the user clicks on a link to a website). 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 somewhat difficult steps to exploit the trust assumptions of programs running on the target system (e.g., the attacker must create a customized 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 system (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, could directly affect a targeted system. The Impact metrics independently reflect the degree of loss of confidentiality, integrity, and availability. For example, exploitation of a particular vulnerability could cause a partial loss of integrity and availability but no loss of confidentiality.

For many misuse vulnerabilities, the possible impact from exploitation is dependent on the privileges held by the user or application (including services). For example, a user account could have restricted privileges or full root-level privileges; exploiting many misuse vulnerabilities would cause a greater impact if full privileges were available. However, the privileges available through exploitation are environment-specific. CMSS prevents environment-specific privileges from affecting the base impact metrics by assuming for those metrics that the generally accepted best practices for the privileges are being employed.[11]

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

2.1.2.1 Confidentiality Impact (C)

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

Table 4. Confidentiality Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the confidentiality of the system.
Partial (P)	There is considerable information disclosure. Access to some system files is possible, but the 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. AND/OR There is considerable unauthorized access to the system. Examples include an authorized user gaining access to certain prohibited system functions, an unauthorized user gaining access to a network service offered by the system, and an unauthorized user gaining user or application-level privileges on the system (such as a database administration account).
Complete (C)	There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system's data (memory, files, etc.) An example is someone who is not authorized to act as a system administrator gaining full administrator privileges to the system.

2.1.2.2 Integrity Impact (I)

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

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

Table 5. Integrity Impact Scoring Evaluation

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

2.1.2.3 Availability Impact (A)

The Availability Impact metric measures the potential impact to availability of a successfully exploited 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. The greater the potential impact, the higher the score.

Table 6. Availability Impact Scoring Evaluation

Metric Value	Description
None (N)	There is no impact to the availability of the system.
Partial (P)	There is reduced performance or interruptions in resource availability. 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 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 it does not apply or the analyst does not have sufficient data to determine the appropriate metric value.

2.2.1 General Exploit Level (GEL)

The General Exploit Level metric measures the prevalence of attacks against a misuse vulnerability—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 frequently exploitation attempts occur for a vulnerability, the higher the score.

Table 7. General Exploit Level Scoring Evaluation

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

2.2.2 General Remediation Level (GRL)

The General Remediation Level metric measures the availability of remediation measures that can mitigate the vulnerability other than rendering the misused software feature useless (e.g., disabling the affected feature, removing the software). Remediation measures may restrict the usage of the feature to minimize or prevent misuse. One example of a remediation measure available against users opening infected email attachments is the antivirus check in an email client that restricts which attachments a user is able to open. Similarly, an antispam or antiphishing 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. Also, users can be made aware of phishing threats and how they should handle emails that could be phishing attacks. 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.

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

Table 8. General Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are available to significantly restrict feature use in such a way as to collectively decrease the incidence of misuse by between 76% and 100%. (A 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 as to collectively decrease the incidence of misuse by between 26% and 75%.
Low (L)	Remediation measures are available to slightly restrict feature use in such a way as to collectively decrease the incidence 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 a particular IT environment. Each organization computing CMSS metrics can determine an appropriate definition of IT environment, such as the entire enterprise or a logical or physical subset of the enterprise (e.g., a facility, a division, a small group of related systems). Since environmental metrics are optional, each includes a metric value that has no effect on the score. This value is used when the analyst feels the particular metric does not apply and wishes to ignore it.

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

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

2.3.1 Local Exploit Level

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

2.3.1.1 Local Vulnerability Prevalence (LVP)

The Local Vulnerability Prevalence metric measures the prevalence of vulnerable systems in a specific environment. It is intended to approximate the percentage of systems that could be affected by the vulnerability. The Local Vulnerability Prevalence depends both on the prevalence of the 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 actions required for vulnerability exploitation. For example, if 80% of the systems contain a particular potentially misused feature but only half of the users 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 approximate percentage of vulnerable systems, the higher the score.

Table 9. Local Vulnerability Prevalence Scoring Evaluation

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

2.3.1.2 Perceived Target Value (PTV)

The Perceived Target Value metric measures the likelihood of attack using the 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 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.

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

Table 11. Local Remediation Level Scoring Evaluation

Metric Value	Description
High (H)	Remediation measures are implemented to restrict feature use in such a way as to collectively decrease the incidence of misuse by between 76% and 100%. (A 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 as to collectively decrease the incidence of misuse by between 26% and 75%.
Low (L)	Remediation measures are implemented to slightly restrict feature use in such a way as to collectively decrease the incidence 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 seven environmental metrics. The first three—Environment Confidentiality Impact, Environment Integrity Impact, and Environment Availability Impact—take the place of the corresponding base impact metrics (Confidentiality Impact, Integrity Impact, and Availability Impact). The fourth, Collateral Damage Potential, augments the three Environment Impact metrics. The remaining three environmental metrics (Confidentiality Requirement, Integrity Requirement, and Availability Requirement) are used to compute scaling factors that are applied to the three Environment Impact metrics.

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

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

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

2.3.3.2 Collateral Damage Potential (CDP)

The Collateral Damage Potential metric measures the potential for loss of life or physical assets through damage or theft of property or equipment. The metric may also measure economic loss of productivity or revenue. This metric can adjust the local impact score to account for application importance. For example, a vulnerability that permits an attacker to gain user-level access to an application (e.g., DNS server, database server) can be scored differently on a system that uses the application in a trivial way versus another system that uses the application in a critical way. The possible values for this metric are listed in Table 12. The greater the damage potential, the higher the vulnerability score. Each organization must determine for itself the precise meaning of “slight,” “moderate,” “significant,” and “catastrophic” in the organization’s environment.

Table 12. Collateral Damage Potential Scoring Evaluation

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

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

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

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

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

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

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

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

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

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

2.4 Base, Temporal, and Environmental Vectors

The CMSS vector facilitates CMSS's open nature. This vector contains the values assigned to each metric, and it is used to communicate exactly how the score for each vulnerability is derived. Therefore, the vector should always be presented with the score.

Each metric in the vector consists of the abbreviated metric name, followed by a ":" (colon), then the abbreviated metric value. The vector lists these metrics in a predetermined order, using the "/" (slash) character to separate the metrics. If a temporal or environmental metric is not to be used, it is given a value of "ND" (not defined). The base, temporal, and environmental vectors are shown below in Table 14.

Table 14. Base, Temporal, and Environmental Vectors

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

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

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

Confidentiality Requirement: Medium, Integrity Requirement: High, Availability Requirement: Low" would produce the following environmental vector: "LVP:H/PTV:M/LRL:L/EC:ND/EI:C/EA:ND/CDP:ND/CR:M/IR:H/AR:L."

3. Scoring

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

3.1 Guidelines

Below are guidelines that should help analysts when scoring misuse vulnerabilities. These guidelines are intended primarily for analysts who 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.

The current versions of CMSS, CCSS, and CVSS are all based on sets of assumptions regarding system security, configuration, use, and management. An example is assuming that a software flaw affecting multiple operating systems should be scored based on the security characteristics of the most widely used of those operating systems. These assumptions are made to simplify the scoring process and ensure consistency among analysts. However, to make CMSS, CCSS, and CVSS more valuable for use in quantitative risk assessment and other complex methodologies, these assumptions should be documented in a standardized way. It is hoped that, in the future, all three specifications will be updated to add a unified convention for documenting these assertions.

3.1.1 General

SCORING TIP #1: Misuse 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 system only.

SCORING TIP #3: When scoring the impact of a vulnerability that has multiple exploitation methods, the analyst should create a vector and calculate a score for each method. The organization using the data would select the appropriate base vector and score for each situation based on the organization's policy and actual system settings.

3.1.2 Base Metrics

3.1.2.1 Access Vector

SCORING TIP #4: When a misuse 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 #5: 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. If misuse caused these to ignore certain types of content that they should be examining, analysts should score the Access Vector of these vulnerabilities as "Network".

3.1.2.2 Authentication

SCORING TIP #6: 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 Authentication 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 would have Access Complexity of “Low” if the credentials are publicized (which is usually the case).

3.1.2.3 Confidentiality, Integrity, Availability Impacts

SCORING TIP #7: Vulnerabilities that give root-level access should be scored with complete loss of confidentiality, integrity, and availability, while vulnerabilities that give lesser degrees of access, such as user-level, should be scored with only partial loss of confidentiality, integrity, and availability. For example, a vulnerability that allows an attacker to modify an operating system password file as desired (which would permit the attacker to change the root password or create a new root-level account) should be scored with complete impact of confidentiality, integrity, and availability. On the other hand, an issue that enables an attacker to impersonate a valid user who has limited privileges should be scored with a partial impact of confidentiality, integrity, and availability.

SCORING TIP #8: Vulnerabilities that permit a partial or complete loss of integrity often also permit an impact to availability. For example, an attacker who is able to modify records can probably also delete them.

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, which the CMSS equations are based on is available at http://www.first.org/cvss/.

3.2.1 Base Equation

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

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

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

Exploitability = 20 * AccessVector * Authentication * AccessComplexity

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

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

```
                low: 0.71
ConfImpact = case ConfidentialityImpact of
                none:         0.0
                partial:      0.275
                complete:     0.660
IntegImpact = case IntegrityImpact of
                none:         0.0
                partial:      0.275
                complete:     0.660
AvailImpact = case AvailabilityImpact of
                none:         0.0
                partial:      0.275
                complete:     0.660
```

The base equation does not include one of the base metrics, Privilege Level. Privilege Level is used by analysts when calculating certain environmental metrics.

3.2.2 Temporal Equation

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

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

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

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

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

3.2.3 Environmental Equation

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

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

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

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

LocalExploitLevel = LocalVulnerabilityPrevalence * PerceivedTargetValue

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

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

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

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

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

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

4. Examples

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

4.1 Example One: ARP Cache Poisoning

The Address Resolution Protocol (ARP) trusts that each ARP reply contains the correct mapping between Media Access Control (MAC) and Internet Protocol (IP) addresses. The ARP cache uses that information to provide a useful service—to enable sending data between devices within a local network. However, 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. The Privilege Level is “Not Defined.”

The base vector is AV:A/AC:L/Au:N/C:N/I:P/A:P/PL:ND. This vector produces an impact subscore of 4.9, an exploitability subscore of 6.5, and a base score of 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”. The temporal vector is GEL:L/GRL:L. The temporal exploitability subscore is 4.1, as opposed to the base exploitability subscore of 6.5, and the temporal score is 3.7, compared to the base score of 4.8. 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 environmental exploitability subscore, 6.2, is higher than the temporal exploitability subscore, 4.1.

Now consider the impact subscore of the environmental score. Suppose that the Collateral Damage Potential in this case is “None”; this metric would not then modify the impact subscore in the environmental score calculation. The organization follows recommended practices, so it sets the three

Environmental Impact metrics to “Not Defined”, which causes no change to the impact subscore. Scores of “Medium” assigned to the Confidentiality Requirement and Availability Requirement also do not modify the impact subscore. However, if the organization gives a score of “High” for the Integrity Requirement because of the importance of integrity in the environment, then the impact subscore will increase because this vulnerability happens to impact integrity. The environmental impact subscore, 6.2, is slightly higher than the base impact subscore of 4.9.

The final environmental score is 5.4. The environmental vector is LVP:H/PTV:M/LRL:N/EC:ND/EI:ND/EA:ND/CDP:N/CR:M/IR:H/AR:M.

4.2 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 by sending 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: an attacker can 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[13]. The Privilege Level is “Not Defined.”

The base vector is AV:N/AC:M/Au:N/C:N/I:P/A:N/PL:ND. This vector produces an impact subscore of 2.9, an exploitability subscore of 8.6, and a base score of 4.3.

Temporal metrics describe the prevalence of attacks against a misuse vulnerability and the 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. The temporal vector is GEL:L/GRL:N. The temporal environmental subscore is 6.9, and the overall temporal score is 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

[13] 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.

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 environmental exploitability subscore, 5.5, is lower than the temporal exploitability subscore, 6.9.

The environmental score also includes an impact subscore. Suppose that the organization scored the Confidentiality Requirement and Integrity Requirement as “Medium,” which do not modify the impact subscore, and the Availability Requirement is rated “Low”. The Low 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”). The organization follows recommended practices and sets the three Environmental Impact metrics to “Not Defined”. Collateral Damage Potential is set to “None” and does not modify the base impact subscore. Since none of these metrics have affected the score, the environmental impact score is 2.9, the same as the base impact subscore.

The final environmental score is 2.8. The environmental vector is LVP:M/PTV:L/LRL:N/EC:ND/EI:ND/EA:ND/CDP:N/CR:M/IR:M/AR:L.

4.3 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. The Privilege Level is “Not Defined.”

The base vector is AV:N/AC:M/Au:N/C:N/I:P/A:N/PL:ND. This vector produces an impact subscore of 2.9, an exploitability subscore of 8.6, and a base score of 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 antiphishing toolbars or maintain blacklists of known phishing sites. The temporal vector is GEL:M/GRL:M. The temporal exploitability subscore is 5.2, and the overall 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 environmental exploitability subscore, 7.4, is higher than the temporal exploitability subscore, 5.2.

The environmental score also includes an impact subscore. Consider an organization that sets the Collateral Damage Potential to "Low" (higher than the default value "None"), the Confidentiality Requirement and Integrity Requirement to "High", and the Availability Requirement to "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 organization follows recommended practices and sets the three Environmental Impact metrics to "Not Defined". The environmental impact subscore is 5.4.

The final environmental score is 5.5. The environmental vector is LVP:H/PTV:H/LRL:M/EC:ND/EI:ND/EA:ND/CDP:L/CR:H/IR:H/AR:M.

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.

5. 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 encouraged to compare the specifications side-by-side. The specifications have similar structures, making such comparisons easy.[14]

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

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

CMSS and CCSS also offer similar sets of environmental metrics, which are considerably more complex than CVSS's metrics. CVSS has three: Collateral Damage Potential, Target Distribution, and Security Requirements. These metrics are all part of CMSS and CCSS as well, although Target Distribution has been renamed Local Vulnerability Prevalence. Two other metrics have been added to CMSS and CCSS: 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 and CCSS also divide their environmental metrics into two groups: Exploitability and Impact. This allows Exploitability and Impact environmental subscores to be generated for CMSS and CCSS; such subscores are not available in CVSS.

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

6. Appendix A—Additional Resources

The following are resources related to CMSS.

- CVSS calculators can be used to calculate base CMSS scores since they use the same metric values and equations. The NIST CVSS calculator can be found at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2.

- The CVSS version 2 specification is available at http://www.first.org/cvss/cvss-guide.html. General information on CVSS's development is documented at http://www.first.org/cvss/.

- NISTIR 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems*, describes the CVSS version 2 specification and also provides insights as to how CVSS scores can be customized for Federal agency-specific purposes. The report is available at http://csrc.nist.gov/publications/PubsNISTIRs.html.

- NISTIR 7502, *The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities*, describes the CCSS specification. The 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	Application Level
A	Availability Impact
AC	Access Complexity
AR	Availability Requirement
ARP	Address Resolution Protocol
Au	Authentication
AV	Access Vector
C	Complete
C	Confidentiality Impact
CCE	Common Configuration Enumeration
CCSS	Common Configuration Scoring System
CDP	Collateral Damage Potential
CMSS	Common Misuse Scoring System
CR	Confidentiality Requirement
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DNS	Domain Name System
EA	Environment Availability Impact
EC	Environment Confidentiality Impact
EI	Environment Integrity Impact
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
HTML	Hypertext Markup Language
I	Integrity Impact
IM	Instant Messaging
IP	Internet Protocol
IR	Integrity Requirement
IR	Interagency Report
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
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency Report
P	Partial
PAM	Pluggable Authentication Module
PL	Privilege Level
PTV	Perceived Target Value
R	Root Level
RFC	Request for Comments
S	Single
U	User Level
URL	Uniform Resource Locator