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

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

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

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

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*

```
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