A Complete Guide to the Common Vulnerability Scoring System Version 2.0

June, 2007

Peter Mell, Karen Scarfone National Institute of Standards and Technology	Sasha Romanosky Carnegie Mellon University

Acknowledgements: The authors sincerely wish to recognize the contributions of all of the CVSS Special Interest Group members, including Barrie Brook, Seth Hanford, Stav Raviv, Gavin Reid, George Theall and Tadashi Yamagishi as well as the authors of the CVSS v1.0 standard [1].

The Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. CVSS consists of three groups: Base, Temporal and Environmental. Each group produces a numeric score ranging from 0 to 10, and a Vector, a compressed textual representation that reflects the values used to derive the score. The Base group represents the intrinsic qualities of a vulnerability. The Temporal group reflects the characteristics of a vulnerability that change over time. The Environmental group represents the characteristics of a vulnerability that are unique to any user's environment. CVSS enables IT managers, vulnerability bulletin providers, security vendors, application vendors and researchers to all benefit by adopting this common language of scoring IT vulnerabilities.

Table of Contents

1 Introduction

Currently, IT management must identify and assess vulnerabilities across many disparate hardware and software platforms. They need to prioritize these vulnerabilities and remediate those that pose the greatest risk. But when there are so many to fix, with each being scored using different scales [2][3][4], how can IT managers convert this mountain of vulnerability data into actionable information? The Common Vulnerability Scoring System (CVSS) is an open framework that addresses this issue. It offers the following benefits:

- **Standardized Vulnerability Scores**: When an organization normalizes vulnerability scores across all of its software and hardware platforms, it can leverage a single vulnerability management policy. This policy may be similar to a service level agreement (SLA) that states how quickly a particular vulnerability must be validated and remediated.
- **Open Framework**: Users can be confused when a vulnerability is assigned an arbitrary score. "Which properties gave it that score? How does it differ from the one released yesterday?" With CVSS, anyone can see the individual characteristics used to derive a score.
- **Prioritized Risk**: When the environmental score is computed, the vulnerability now becomes contextual. That is, vulnerability scores are now representative of the actual risk to an organization. Users know how important a given vulnerability is in relation to other vulnerabilities.

1.1 What is CVSS?

CVSS is composed of three metric groups: Base, Temporal, and Environmental, each consisting of a set of metrics, as shown in Figure 1.

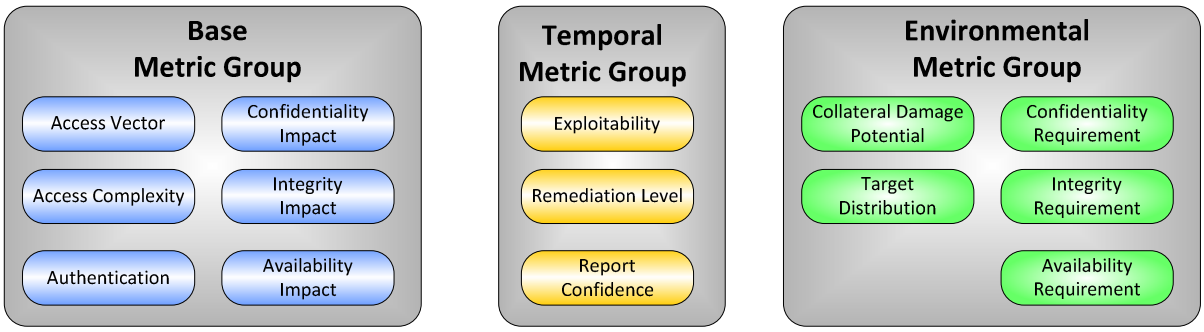

Figure 1: CVSS Metric Groups

These metric groups are described as follows:

- **Base**: represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. Base metrics are discussed in Section 2.1.

- **Temporal**: represents the characteristics of a vulnerability that change over time but not among user environments. Temporal metrics are discussed in Section 2.2.
- **Environmental**: represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. Environmental metrics are discussed in Section 2.3.

The purpose of the CVSS base group is to define and communicate the fundamental characteristics of a vulnerability. This objective approach to characterizing vulnerabilities provides users with a clear and intuitive representation of a vulnerability. Users can then invoke the temporal and environmental groups to provide contextual information that more accurately reflects the risk *to their unique environment*. This allows them to make more informed decisions when trying to mitigate risks posed by the vulnerabilities.

1.2 Other vulnerability scoring systems

There are a number of other vulnerability "scoring" systems managed by both commercial and non-commercial organizations. They each have their merits, but they differ by what they measure. For example, CERT/CC produces a numeric score ranging from 0 to 180 but considers such factors as whether the Internet infrastructure is at risk and what sort of preconditions are required to exploit the vulnerability [3]. The SANS vulnerability analysis scale considers whether the weakness is found in default configurations or client or server systems [4]. Microsoft's proprietary scoring system tries to reflect the difficulty of exploitation and the overall impact of the vulnerability [2]. While useful, these scoring systems provide a one-size-fits-all approach by assuming that the impact for a vulnerability is constant for every individual and organization.

CVSS can also be described by what it is not. That is, it is none of the following:

- A threat rating system such as those used by the US Department of Homeland Security, and the Sans Internet Storm Center.[1] These services provide an advisory warning system for threats to critical US and global IT networks, respectively.
- A vulnerability database such as the National Vulnerability Database (NVD), Open Source Vulnerability Database (OSVDB) or Bugtraq. These databases provide a rich catalogue of known vulnerabilities and vulnerability details.
- A vulnerability identification system such as the industry-standard Common Vulnerabilities and Exposures (CVE) or a weakness dictionary such as the Common Weakness Enumeration (CWE). These frameworks are meant to uniquely identify and classify vulnerabilities according to the causes "as they are manifested in code, design, or architecture."[2]

1.3 How does CVSS work?

When the base metrics are assigned values, the base equation calculates a score ranging from 0 to 10, and creates a vector, as illustrated below in Figure 2. The vector facilitates the "open" nature of the framework. It is a text string that 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 displayed with the vulnerability score. Vectors are further explained in Section 2.4.

[1] http://www.dhs.gov/xinfoshare/programs/Copy_of_press_release_0046.shtm, http://isc.sans.org/
[2] http://cve.mitre.org/, http://cwe.mitre.org/index.html , http://cwe.mitre.org/about/process.html

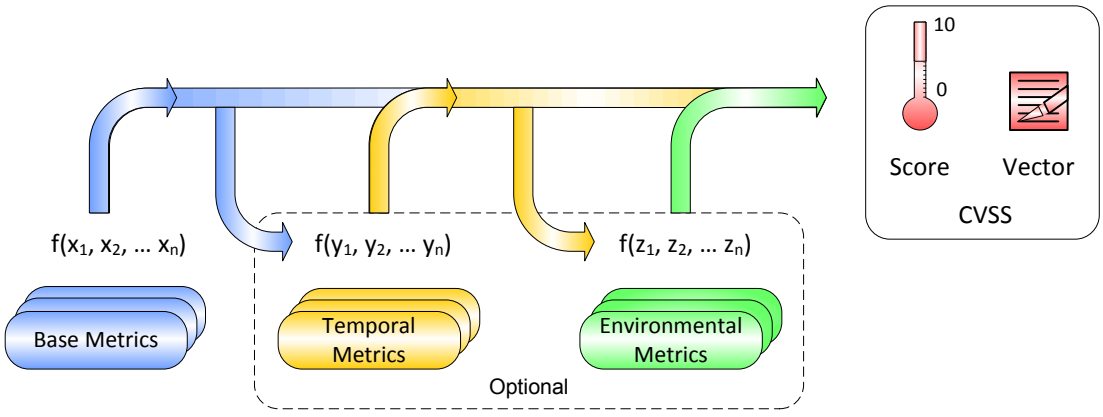

Figure 2: CVSS Metrics and Equations

Optionally, the base score can be refined by assigning values to the temporal and environmental metrics. This is useful in order to provide additional context for a vulnerability by more accurately reflecting the risk posed by the vulnerability to a user's environment. However, this is not required. Depending on one's purpose, the base score and vector may be sufficient.

If a temporal score is needed, the temporal equation will combine the temporal metrics with the base score to produce a temporal score ranging from 0 to 10. Similarly, if an environmental score is needed, the environmental equation will combine the environmental metrics with the temporal score to produce an environmental score ranging from 0 to 10. Base, temporal and environmental equations are fully described in Section 3.2.

1.4 Who performs the scoring?

Generally, the base and temporal metrics are specified by vulnerability bulletin analysts, security product vendors, or application vendors because they typically have better information about the characteristics of a vulnerability than do users. The environmental metrics, however, are specified by users because they are best able to assess the potential impact of a vulnerability within their own environments.

1.5 Who owns CVSS?

CVSS is under the custodial care of the Forum of Incident Response and Security Teams (FIRST).[3] However, it is a completely free and open standard. No organization "owns" CVSS and membership in FIRST is not required to use or implement CVSS. Our only request is that those organizations who publish scores conform to the guidelines described in this document and provide both the score and the scoring vector (described below) so others can understand how the score was derived.

1.6 Who is using CVSS?

[3] www.first.org/cvss

Many organizations are using CVSS, and each are finding value in different ways. Below are some examples:

- **Vulnerability Bulletin Providers:** Both non-profit and commercial organizations are publishing CVSS base and temporal scores and vectors in their free vulnerability bulletins. These bulletins offer much information, including the date of discovery, systems affected and links to vendors for patching recommendations.
- **Software Application Vendors:** Software application vendors are providing CVSS base scores and vectors to their customers. This helps them properly communicate the severity of vulnerabilities in their products and helps their customers effectively manage their IT risk.
- **User Organizations:** Many private-sector organizations are using CVSS internally to make informed vulnerability management decisions. They use scanners or monitoring technologies to first locate host and application vulnerabilities. They combine this data with CVSS base, temporal and environmental scores to obtain more contextual risk information and remediate those vulnerabilities that pose the greatest risk to their systems.
- **Vulnerability Scanning and Management:** Vulnerability management organizations scan networks for IT vulnerabilities. They provide CVSS base scores for every vulnerability on each host. User organizations use this critical data stream to more effectively manage their IT infrastructures by reducing outages and protecting against malicious and accidental IT threats.
- **Security (Risk) Management**: Security Risk Management firms use CVSS scores as input to calculating an organization's risk or threat level. These firms use sophisticated applications that often integrate with an organization's network topology, vulnerability data, and asset database to provide their customers with a more informed perspective of their risk level.
- **Researchers**: The open framework of CVSS enables researchers to perform statistical analysis on vulnerabilities and vulnerability properties.

1.7 Quick definitions

Throughout this document the following definitions are used:

- **Vulnerability**: a bug, flaw, weakness, or exposure of an application, system, device, or service that could lead to a failure of confidentiality, integrity, or availability.
- **Threat**: the likelihood or frequency of a harmful event occurring.
- **Risk**: the relative impact that an exploited vulnerability would have to a user's environment.

2 Metric Groups

2.1 Base Metrics

The base metric group captures the characteristics of a vulnerability that are constant with time and across user environments. The Access Vector, Access Complexity, and Authentication metrics capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it. The three impact metrics measure how a vulnerability, if exploited, will directly affect an IT asset, where the impacts are independently defined as the degree of loss of confidentiality, integrity, and availability. For example, a vulnerability could cause a partial loss of integrity and availability, but no loss of confidentiality.

2.1.1 Access Vector (AV)

This metric reflects how the vulnerability is exploited. The possible values for this metric are listed in Table 1. The more remote an attacker can be to attack a host, the greater the vulnerability score.

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. Examples of locally exploitable vulnerabilities are peripheral attacks such as Firewire/USB DMA attacks, and local privilege escalations (e.g., sudo).
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.
Network (N)	A vulnerability exploitable with *network access* means the vulnerable software is bound to the network stack and the attacker does not require local network access or local access. Such a vulnerability is often termed "remotely exploitable". An example of a network attack is an RPC buffer overflow.

Table 1: Access Vector Scoring Evaluation

2.1.2 Access Complexity (AC)

This metric measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. For example, consider a buffer overflow in an Internet service: once the target system is located, the attacker can launch an exploit at will.

Other vulnerabilities, however, may require additional steps in order to be exploited. For example, a vulnerability in an email client is only exploited after the user downloads and opens a tainted attachment. The possible values for this metric are listed in Table 2. The lower the required complexity, the higher the vulnerability score.

Metric Value	Description
High (H)	Specialized access conditions exist. For example: • In most configurations, the attacking party must already have elevated privileges or spoof additional systems in addition to the attacking system (e.g., DNS hijacking). • The attack depends on social engineering methods that would be easily detected by knowledgeable people. For example, the victim must perform several suspicious or atypical actions. • The vulnerable configuration is seen very rarely in practice. • If a race condition exists, the window is very narrow.
Medium (M)	The access conditions are somewhat specialized; the following are examples: • The attacking party is limited to a group of systems or users at some level of authorization, possibly untrusted. • Some information must be gathered before a successful attack can be launched. • The affected configuration is non-default, and is not commonly configured (e.g., a vulnerability present when a server performs user account authentication via a specific scheme, but not present for another authentication scheme).

	• The attack requires 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).
Low (L)	Specialized access conditions or extenuating circumstances do not exist. The following are examples: • The affected product typically requires access to a wide range of systems and users, possibly anonymous and untrusted (e.g., Internet-facing web or mail server). • The affected configuration is default or ubiquitous. • The attack can be performed manually and requires little skill or additional information gathering. • The "race condition" is a lazy one (i.e., it is technically a race but easily winnable).

Table 2: Access Complexity Scoring Evaluation

2.1.3 Authentication (Au)

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

It is important to note that the Authentication metric is different from Access Vector. Here, authentication requirements are considered *once the system has already been accessed*. Specifically, for locally exploitable vulnerabilities, this metric should only be set to "single" or "multiple" if authentication is needed beyond what is required to log into the system. An example of a locally exploitable vulnerability that requires authentication is one affecting a database engine listening on a Unix domain socket (or some other non-network interface). If the user must authenticate as a valid database user in order to exploit the vulnerability, then this metric should be set to "single."

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.

Table 3: Authentication Scoring Evaluation

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

2.1.4 Confidentiality Impact (C)

This metric measures the impact on confidentiality of a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. The possible values for this metric are listed in Table 4. Increased confidentiality impact increases the vulnerability score.

Metric Value	Description
None (N)	There is no impact to the confidentiality of the system.
Partial (P)	There is considerable informational disclosure. Access to some system files is possible, but the attacker does not have control over what is obtained, or the scope of the loss is constrained. An example is a vulnerability that divulges only certain tables in a database.
Complete (C)	There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system's data (memory, files, etc.)

Table 4: Confidentiality Impact Scoring Evaluation

2.1.5 Integrity Impact (I)

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

Metric Value	Description
None (N)	There is no impact to the integrity of the system.
Partial (P)	Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited. For example, system or application files may be overwritten or modified, but either the attacker has no control over which files are affected or the attacker can modify files within only a limited context or scope.
Complete (C)	There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised. The attacker is able to modify any files on the target system.

Table 5: Integrity Impact Scoring Evaluation

2.1.6 Availability Impact (A)

This metric measures the impact to availability of a successfully exploited vulnerability. Availability refers to the accessibility of information resources. Attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of a system. The possible values for this metric are listed in Table 6. Increased availability impact increases the vulnerability score.

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.

Table 6: Availability Impact Scoring Evaluation

2.2 Temporal Metrics

The threat posed by a vulnerability may change over time. Three such factors that CVSS captures are: confirmation of the technical details of a vulnerability, the remediation status of the vulnerability, and the availability of exploit code or techniques. Since temporal metrics are optional they each include a metric value that has no effect on the score. This value is used when the user feels the particular metric does not apply and wishes to "skip over" it.

2.2.1 Exploitability (E)

This metric measures the current state of exploit techniques or 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.

Initially, real-world exploitation may only be theoretical. Publication of proof of concept code, functional exploit code, or sufficient technical details necessary to exploit the vulnerability may follow. Furthermore, the exploit code available may progress from a proof-of-concept demonstration to exploit code that is successful in exploiting the vulnerability consistently. In severe cases, it may be delivered as the payload of a network-based worm or virus. The possible values for this metric are listed in Table 7. The more easily a vulnerability can be exploited, the higher the vulnerability score.

Metric Value	Description
Unproven (U)	No exploit code is available, or an exploit is entirely theoretical.
Proof-of-Concept (POC)	Proof-of-concept exploit code or an attack demonstration that is not practical for most systems is available. The code or technique is not functional in all situations and may require substantial modification by a skilled attacker.
Functional (F)	Functional exploit code is available. The code works in most situations where the vulnerability exists.
High (H)	Either the vulnerability is exploitable by functional mobile autonomous code, or no exploit is required (manual trigger) and details are widely available. The code works in every situation, or is actively being delivered via a mobile autonomous agent (such as a worm or virus).
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.

Table 7: Exploitability Scoring Evaluation

2.2.2 Remediation Level (RL)

The remediation level of a vulnerability is an important factor for prioritization. The typical vulnerability is unpatched when initially published. Workarounds or hotfixes may offer interim remediation until an official patch or upgrade is issued. Each of these respective stages adjusts the temporal score downwards, reflecting the decreasing urgency as remediation becomes final. The possible values for this metric are listed in Table 8. The less official and permanent a fix, the higher the vulnerability score is.

Metric Value	Description
Official Fix (OF)	A complete vendor solution is available. Either the vendor has issued an official patch, or an upgrade is available.
Temporary Fix	There is an official but temporary fix available. This includes instances where the

(TF)	vendor issues a temporary hotfix, tool, or workaround.
Workaround (W)	There is an unofficial, non-vendor solution available. In some cases, users of the affected technology will create a patch of their own or provide steps to work around or otherwise mitigate the vulnerability.
Unavailable (U)	There is either no solution available or it is impossible to apply.
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.

Table 8: Remediation Level Scoring Evaluation

2.2.3 Report Confidence (RC)

This metric measures the degree of confidence in the existence of the vulnerability and the credibility of the known technical details. Sometimes, only the existence of vulnerabilities are publicized, but without specific details. The vulnerability may later be corroborated and then confirmed through acknowledgement by the author or vendor of the affected technology. The urgency of a vulnerability is higher when a vulnerability is known to exist with certainty. This metric also suggests the level of technical knowledge available to would-be attackers. The possible values for this metric are listed in Table 9. The more a vulnerability is validated by the vendor or other reputable sources, the higher the score.

Metric Value	Description
Unconfirmed (UC)	There is a single unconfirmed source or possibly multiple conflicting reports. There is little confidence in the validity of the reports. An example is a rumor that surfaces from the hacker underground.
Uncorroborated (UR)	There are multiple non-official sources, possibly including independent security companies or research organizations. At this point there may be conflicting technical details or some other lingering ambiguity.
Confirmed (C)	The vulnerability has been acknowledged by the vendor or author of the affected technology. The vulnerability may also be "Confirmed" when its existence is confirmed from an external event such as publication of functional or proof-of-concept exploit code or widespread exploitation.
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.

Table 9: Report Confidence Scoring Evaluation

2.3 Environmental Metrics

Different environments can have an immense bearing on the risk that a vulnerability poses to an organization and its stakeholders. The CVSS environmental metric group captures the characteristics of a vulnerability that are associated with a user's IT environment. Since environmental metrics are optional they each include a metric value that has no effect on the score. This value is used when the user feels the particular metric does not apply and wishes to "skip over" it.

2.3.1 Collateral Damage Potential (CDP)

This 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. The possible values for this metric are listed in Table 10. Naturally, the greater the damage potential, the higher the vulnerability score.

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

Table 10: Collateral Damage Potential Scoring Evaluation

Clearly, each organization must determine for themselves the precise meaning of "slight, moderate, significant, and catastrophic."

2.3.2 Target Distribution (TD)

This metric measures the proportion of vulnerable systems. It is meant as an environment-specific indicator in order to approximate the percentage of systems that could be affected by the vulnerability. The possible values for this metric are listed in Table 11. The greater the proportion of vulnerable systems, the higher the score.

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% - 25% of the total environment is at risk.
Medium (M)	Targets exist inside the environment, but on a medium scale. Between 26% - 75% of the total environment is at risk.
High (H)	Targets exist inside the environment on a considerable scale. Between 76% - 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.

Table 11: Target Distribution Scoring Evaluation

2.3.3 Security Requirements (CR, IR, AR)

These metrics enable the analyst to customize the CVSS score depending on the importance of the affected IT asset to a user's organization, measured in terms of confidentiality, integrity, and availability, That is, if an IT asset supports a business function for which availability is most important, the analyst can assign a greater value to availability, relative to confidentiality and integrity. Each security requirement has three possible values: "low," "medium," or "high."

The full effect on the environmental score is determined by the corresponding base impact metrics. That is, these metrics modify the environmental score by reweighting the (base) confidentiality, integrity, and

availability impact metrics.[4] For example, the confidentiality impact (C) metric has *increased* weight if the confidentiality requirement (CR) is "high." Likewise, the confidentiality impact metric has *decreased* weight if the confidentiality requirement is "low." The confidentiality impact metric weighting is neutral if the confidentiality requirement is "medium." This same logic is applied to the integrity and availability requirements.

Note that the confidentiality requirement will not affect the environmental score if the (base) confidentiality impact is set to "none." Also, increasing the confidentiality requirement from "medium" to "high" will not change the environmental score when the (base) impact metrics are set to "complete." This is because the impact sub score (part of the base score that calculates impact) is already at a maximum value of 10.

The possible values for the security requirements are listed in Table 12. For brevity, the same table is used for all three metrics. The greater the security requirement, the higher the score (remember that "medium" is considered the default). These metrics will modify the score as much as plus or minus 2.5.

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.

Table 12: Security Requirements Scoring Evaluation

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.[5] CVSS follows this general model of FIPS 199, but does not require organizations to use any particular system for assigning the low, medium, and high impact ratings.

2.4 Base, Temporal, Environmental Vectors

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

[4] Please note that the base confidentiality, integrity and availability impact metrics, themselves, are not changed.
[5] http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

value of “ND” (not defined). The base, temporal, and environmental vectors are shown below in Table 13.

Metric Group	Vector
Base	AV:[L,A,N]/AC:[H,M,L]/Au:[M,S,N]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C]
Temporal	E:[U,POC,F,H,ND]/RL:[OF,TF,W,U,ND]/RC:[UC,UR,C,ND]
Environmental	CDP:[N,L,LM,MH,H,ND]/TD:[N,L,M,H,ND]/CR:[L,M,H,ND]/ IR:[L,M,H,ND]/AR:[L,M,H,ND]

Table 13: Base, Temporal and Environmental Vectors

For example, a vulnerability with base metric values of “Access Vector: Low, Access Complexity: Medium, Authentication: None, Confidentiality Impact: None, Integrity Impact: Partial, Availability Impact: Complete” would have the following base vector: “AV:L/AC:M/Au:N/C:N/I:P/A:C.”

3 Scoring

3.1 Guidelines

Below are guidelines that should help analysts when scoring vulnerabilities.

3.1.1 General

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

SCORING TIP #2: When scoring a vulnerability, consider the direct impact to the target host only. For example, consider a cross-site scripting vulnerability: the impact to a user’s system could be much greater than the impact to the target host. However, this is an indirect impact. Cross-site scripting vulnerabilities should be scored with no impact to confidentiality or availability, and partial impact to integrity.

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

SCORING TIP #4: When scoring the impact of a vulnerability that has multiple exploitation methods (attack vectors), the analyst should choose the exploitation method that causes the greatest impact, rather than the method which is most common, or easiest to perform. For example, if functional exploit code exists for one platform but not another, then Exploitability should be set to “Functional”. If two separate variants of a product are in parallel development (e.g. PHP 4.x and PHP 5.x), and a fix exists for one variant but not another, then the Remediation Level should be set to “Unavailable”.

3.1.2 Base Metrics

3.1.2.1 Access Vector

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

SCORING TIP #6: Many client applications and utilities have local vulnerabilities that can be exploited remotely either through user-complicit actions or via automated processing. For example, decompression utilities and virus scanners automatically scan incoming email messages. Also, helper applications (office suites, image viewers, media players, etc.) are exploited when malicious files are exchanged via e-mail or downloaded from web sites. Therefore, analysts should score the Access Vector of these vulnerabilities as "Network".

3.1.2.2 Authentication

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

3.1.2.3 Confidentiality, Integrity, Availability Impacts

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

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

3.2 Equations

Scoring equations and algorithms for the base, temporal and environmental metric groups are described below. Further discussion of the origin and testing of these equations is available at www.first.org/cvss.

3.2.1 Base Equation

The base equation is the foundation of CVSS scoring. The base equation is:

```
BaseScore[6] = 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*AccessComplexity*Authentication
```

[6] This is formula version 2.10

```
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

AccessComplexity = case AccessComplexity of
                       high: 0.35
                       medium: 0.61
                       low: 0.71

Authentication   = case Authentication of
                       requires multiple instances of authentication: 0.45
                       requires single instance of authentication: 0.56
                       requires no authentication: 0.704

ConfImpact       = case ConfidentialityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

IntegImpact      = case IntegrityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

AvailImpact      = case AvailabilityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660
```

3.2.2 Temporal Equation

If employed, the temporal equation will combine the temporal metrics with the base score to produce a temporal score ranging from 0 to 10. Further, the temporal score will produce a temporal score no higher than the base score, and no less than 33% lower than the base score. The temporal equation is:

```
TemporalScore = round_to_1_decimal(BaseScore*Exploitability
                *RemediationLevel*ReportConfidence)

Exploitability   = case Exploitability of
                       unproven:             0.85
                       proof-of-concept:     0.9
                       functional:           0.95
                       high:                 1.00
                       not defined:          1.00

RemediationLevel = case RemediationLevel of
                       official-fix:         0.87
                       temporary-fix:        0.90
                       workaround:           0.95
                       unavailable:          1.00
                       not defined:          1.00

ReportConfidence = case ReportConfidence of
                       unconfirmed:          0.90
```

```
                    uncorroborated:       0.95
                    confirmed:            1.00
                    not defined:          1.00
```

3.2.3 Environmental Equation

If employed, the environmental equation will combine the environmental metrics with the temporal score to produce an environmental score ranging from 0 to 10. Further, this equation will produce a score no higher than the temporal score. The environmental equation is:

```
EnvironmentalScore = round_to_1_decimal((AdjustedTemporal+
(10-AdjustedTemporal)*CollateralDamagePotential)*TargetDistribution)

AdjustedTemporal = TemporalScore recomputed with the BaseScore's Impact sub-
equation replaced with the AdjustedImpact equation

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

CollateralDamagePotential = case CollateralDamagePotential of
                                 none:            0
                                 low:             0.1
                                 low-medium:      0.3
                                 medium-high:     0.4
                                 high:            0.5
                                 not defined:     0

TargetDistribution        = case TargetDistribution of
                                 none:            0
                                 low:             0.25
                                 medium:          0.75
                                 high:            1.00
                                 not defined:     1.00

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
```

3.3 Examples

Below, we provide examples of how CVSS is used for three different vulnerabilities.

3.3.1 CVE-2002-0392

Consider CVE-2002-0392: Apache Chunked-Encoding Memory Corruption Vulnerability. In June 2002, a vulnerability was discovered in the means by which the Apache web server handles requests encoded using chunked encoding. The Apache Foundation reported that a successful exploit can lead to denial of service in some cases, and in others, the execution of arbitrary code with the privileges of the web server.

Since the vulnerability can be exploited remotely, the Access Vector is "Network". The Access Complexity is "Low" because no additional circumstances need to exist for this exploit to be successful; the attacker need only craft a proper exploit message to the Apache web listener. No authentication is required to trigger the vulnerability (any Internet user can connect to the web server), so the Authentication metric is "None".

Since the vulnerability can be exploited using multiple methods with different outcomes, scores need to be generated for each method and the highest used.

If the vulnerability is exploited to execute arbitrary code with the permissions of the web server, thereby altering web content and possibly viewing local user or configuration information (including connection settings and passwords to back-end databases), the Confidentiality and Integrity Impact metrics are set to “Partial”. Together, these metrics result in a base score of 6.4.

If the vulnerability is exploited to cause a denial of service, the Availability Impact is set to “Complete”. Together, the metrics produce a base score of 7.8. Since this is the highest possible base score of the exploitation options, it is used as the base score.

The base vector for this vulnerability is therefore: AV:N/AC:L/Au:N/C:N/I:N/A:C.

Exploit code is known to exist and therefore Exploitability is set to “Functional”. The Apache foundation has released patches for this vulnerability (available to both 1.3 and 2.0) and so Remediation Level is “Official-Fix”. Naturally, report confidence is “Confirmed”. These metrics adjust the base score to give a temporal score of 6.4.

Assuming that availability is more important than usual for the targeted systems, and depending on the values for Collateral Damage Potential and Target Distribution, the environmental score could vary between 0.0 (“None”, “None”) and 9.2 (“High”, “High”). The results are summarized below.

```
-----------------------------------------------------
BASE METRIC                 EVALUATION         SCORE
-----------------------------------------------------
Access Vector               [Network]         (1.00)
Access Complexity           [Low]             (0.71)
Authentication              [None]            (0.704)
Confidentiality Impact      [None]            (0.00)
Integrity Impact            [None]            (0.00)
Availability Impact         [Complete]        (0.66)
-----------------------------------------------------
BASE FORMULA                                BASE SCORE
-----------------------------------------------------
Impact = 10.41*(1-(1)*(1)*(0.34)) == 6.9
Exploitability = 20*0.71*0.704*1 == 10.0
f(Impact) = 1.176
BaseScore = (0.6*6.9 + 0.4*10.0 – 1.5)*1.176
```

```
                                                  == (7.8)
-----------------------------------------------------

-----------------------------------------------------
TEMPORAL METRIC             EVALUATION         SCORE
-----------------------------------------------------
Exploitability              [Functional]      (0.95)
Remediation Level           [Official-Fix]    (0.87)
Report Confidence           [Confirmed]       (1.00)
-----------------------------------------------------
TEMPORAL FORMULA                         TEMPORAL SCORE
-----------------------------------------------------
round(7.8 * 0.95 * 0.87 * 1.00)             == (6.4)
-----------------------------------------------------

-----------------------------------------------------
ENVIRONMENTAL METRIC        EVALUATION         SCORE
-----------------------------------------------------
Collateral Damage Potential [None - High]  {0 - 0.5}
Target Distribution         [None - High]  {0 - 1.0}
Confidentiality Req.        [Medium]           (1.0)
Integrity Req.              [Medium]           (1.0)
Availability Req.           [High]             (1.51)
-----------------------------------------------------
ENVIRONMENTAL FORMULA            ENVIRONMENTAL SCORE
-----------------------------------------------------
AdjustedImpact = min(10,10.41*(1-(1-0*1)*(1-0*1)
        *(1-0.66*1.51))                     == (10.0)
AdjustedBase =((0.6*10)+(0.4*10.0)–1.5)*1.176
                                            == (10.0)
AdjustedTemporal == (10*0.95*0.87*1.0)       == (8.3)
EnvScore = round((8.3+(10-8.3)*{0-0.5})*{0-1})
                                    == (0.00 - 9.2)
-----------------------------------------------------
```

3.3.2 CVE-2003-0818

Consider CVE-2003-0818: Microsoft Windows ASN.1 Library Integer Handling Vulnerability. In September 2003, a vulnerability was discovered that targets the ASN.1 library of all Microsoft operating systems. Successful exploitation of this vulnerability results in a buffer overflow condition allowing the attacker to execute arbitrary code with administrative (system) privileges.

This is a remotely exploitable vulnerability that does not require authentication, therefore the Access Vector is “Network” and “Authentication” is “None”. The Access Complexity is “Low” because no additional access or specialized circumstances need to exist for the exploit to be successful. Each of the Impact metrics is set to “Complete” because of the possibility of a complete system compromise. Together, these metrics produce a maximum base score of 10.0.

The base vector for this vulnerability is therefore: AV:N/AC:L/Au:N/C:C/I:C/A:C.

Known exploits do exist for this vulnerability and so Exploitability is “Functional”. In February 2004, Microsoft released patch MS04-007, making the Remediation Level “Official-Fix” and the Report Confidence “Confirmed”. These metrics adjust the base score to give a temporal score of 8.3.

Assuming that availability is less important than usual for the targeted systems, and depending on the values for Collateral Damage Potential and Target Distribution, the environmental score could vary between 0.0 (“None”, “None”) and 9.0 (“High”, “High”). The results are summarized below.

```
-----------------------------------------------------
BASE METRIC                 EVALUATION         SCORE
-----------------------------------------------------
Access Vector               [Network]         (1.00)
Access Complexity           [Low]             (0.71)
Authentication              [None]            (0.704)
Confidentiality Impact      [Complete]        (0.66)
Integrity Impact            [Complete]        (0.66)
Availability Impact         [Complete]        (0.66)
-----------------------------------------------------
FORMULA                                    BASE SCORE
-----------------------------------------------------
Impact = 10.41*(1-(0.34*0.34*0.34)) == 10.0
Exploitability = 20*0.71*0.704*1 == 10.0
f(Impact) = 1.176
BaseScore =((0.6*10.0)+(0.4*10.0)–1.5)*1.176
                                             == (10.0)
-----------------------------------------------------

-----------------------------------------------------
TEMPORAL METRIC             EVALUATION         SCORE
-----------------------------------------------------
Exploitability              [Functional]      (0.95)
Remediation Level           [Official-Fix]    (0.87)
Report Confidence           [Confirmed]       (1.00)
-----------------------------------------------------
FORMULA                                 TEMPORAL SCORE
-----------------------------------------------------
round(10.0 * 0.95 * 0.87 * 1.00) ==            (8.3)
-----------------------------------------------------

-----------------------------------------------------
ENVIRONMENTAL METRIC        EVALUATION         SCORE
-----------------------------------------------------
Collateral Damage Potential [None - High]  {0 - 0.5}
Target Distribution         [None - High]  {0 - 1.0}
Confidentiality Req.        [Medium]           (1.0)
Integrity Req.              [Medium]           (1.0)
Availability Req.           [Low]              (0.5)
-----------------------------------------------------
FORMULA                           ENVIRONMENTAL SCORE
-----------------------------------------------------
AdjustedImpact = 10.41*(1-(1-0.66*1)*(1-0.66*1)
         *(1-0.66*0.5)) == 9.6
AdjustedBase =((0.6*9.6)+(0.4*10.0)–1.5)*1.176
                                               == (9.7)
AdjustedTemporal == (9.7*0.95*0.87*1.0)     == (8.0)
EnvScore = round((8.0+(10-8.0)*{0-0.5})*{0-1})
                 ==                       (0.00 - 9.0)
-----------------------------------------------------
```

3.3.3 CVE-2003-0062

Consider CVE-2003-0062: Buffer Overflow in NOD32 Antivirus. NOD32 is an antivirus software application developed by Eset. In February 2003, a buffer overflow vulnerability was discovered in Linux and Unix versions prior to 1.013 that could allow local users to execute arbitrary code with the privileges of the user executing NOD32. To trigger the buffer overflow, the attacker must wait for (or coax) another user (possibly root) to scan a directory path of excessive length.

Since the vulnerability is exploitable only to a user locally logged into the system, the Access Vector is "Local". The Access Complexity is "High" because this vulnerability is not exploitable at the attacker's whim. There is an additional layer of complexity because the attacker must wait for another user to run the virus scanning software. Authentication is set to "None" because the attacker does not need to authenticate to any additional system. If an administrative user were to run the virus scan, causing the buffer overflow, then a full system compromise would be possible. Since the most harmful case must be considered, each of the three Impact metrics is set to "Complete". Together, these metrics produce a base score of 6.2.

The base vector for this vulnerability is therefore: AV:L/AC:H/Au:N/C:C/I:C/A:C.

Partial exploit code has been released, so the Exploitability metric is set to "Proof-Of-Concept". Eset has released updated software, giving a Remediation Level of "Official-Fix" and Report Confidence of "Confirmed". These three metrics adjust the base score to give a temporal score of 4.9.

Assuming that confidentiality, integrity, and availability are roughly equally important for the targeted systems, and depending on the values for Collateral Damage Potential and Target Distribution, the environmental score could vary between 0.0 ("None", "None") and 7.5 ("High", "High"). The results are summarized below.

BASE METRIC	EVALUATION	SCORE
Access Vector	[Local]	(0.395)
Access Complexity	[High]	(0.35)
Authentication	[None]	(0.704)
Confidentiality Impact	[Complete]	(0.66)
Integrity Impact	[Complete]	(0.66)
Availability Impact	[Complete]	(0.66)

FORMULA	BASE SCORE
Impact = 10.41*(1-(0.34*0.34*0.34)) == 10.0	
Exploitability = 20*0.35*0.704*0.395 == 1.9	
f(Impact) = 1.176	
BaseScore =((0.6*10)+(0.4*1.9)–1.5)*1.176	
	== (6.2)

TEMPORAL METRIC	EVALUATION	SCORE
Exploitability	[Proof-Of-Concept]	(0.90)
Remediation Level	[Official-Fix]	(0.87)
Report Confidence	[Confirmed]	(1.00)

FORMULA	TEMPORAL SCORE
round(6.2 * 0.90 * 0.87 * 1.00) ==	(4.9)

```
--------------------------------------------------

--------------------------------------------------
ENVIRONMENTAL METRIC        EVALUATION         SCORE
--------------------------------------------------
Collateral Damage Potential [None - High]  {0 - 0.5}
Target Distribution         [None - High]  {0 - 1.0}
Confidentiality Req.        [Medium]           (1.0)
Integrity Req.              [Medium]           (1.0)
Availability Req.           [Medium]           (1.0)
--------------------------------------------------
FORMULA                           ENVIRONMENTAL SCORE
--------------------------------------------------
AdjustedTemporal == 4.9
EnvScore = round((4.9+(10-4.9)*{0-0.5})*{0-1})
                                      ==  (0.00 - 7.5)
--------------------------------------------------
```

4 Additional Resources

Below, we present a list of resources that may be useful to anyone implementing CVSS. Vulnerability bulletins are helpful when searching for detailed information about a particular vulnerability. CVSS calculators are helpful when trying to compute your own base, temporal or environmental scores.

Vulnerability bulletins:

- The National Institute of National Institute of Standards and Technology (NIST) maintains the National Vulnerability Database (NVD), a vulnerability bulletin website that includes CVSS base scores. NIST provides these web-based bulletins in addition to XML feeds free for use. They can be found at http://nvd.nist.gov/nvd.cfm, and http://nvd.nist.gov/download.cfm#XML, respectively.

- IBM Internet Security Systems (ISS) publishes X-Force vulnerability bulletins free for use. They include CVSS base and temporal scores and can be found at http://xforce.iss.net/xforce/alerts.

- Qualys publishes vulnerability references that include both CVSS base and temporal scores. These can be found at http://www.qualys.com/research/alerts/.

- Cisco vulnerability bulletins including CVSS base and temporal scores can be found at http://tools.cisco.com/MySDN/Intelligence/home.x. (Note: requires a Cisco Connection Online account).

- Tenable Network Security publishes plugins for the Nessus vulnerability scanning tool. These plugins that include CVSS base score can be found at http://www.nessus.org/plugins/.

CVSS Calculators

- The NIST CVSSv2 calculator can be found at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2

- The Information-Technology Promotion Agency of Japan: http://jvnrss.ise.chuo-u.ac.jp/jtg/cvss/en/CVSSv2.html

5 Final Remarks

The authors recognize that many other metrics could have been included in CVSS. We also realize that no one scoring system will fit everyone's needs perfectly. The particular metrics used in CVSS were identified as the best compromise between completeness, ease-of-use and accuracy. They represent the cumulative experience of the CVSS Special Interest Group members as well as extensive testing of real-world vulnerabilities in end-user environments. As CVSS matures, these metrics may expand or adjust, making the scoring even more accurate, flexible and representative of modern vulnerabilities and their risks.

6 References

[1] Mike Schiffman, Gerhard Eschelbeck, David Ahmad, Andrew Wright, Sasha Romanosky, "CVSS: A Common Vulnerability Scoring System", National Infrastructure Advisory Council (NIAC), 2004.

[2] Microsoft Corporation. Microsoft Security Response Center Security Bulletin Severity Rating System. November 2002 [cited 16 March 2007]. Available from URL: http://www.microsoft.com/technet/security/bulletin/rating.mspx.

[3] United States Computer Emergency Readiness Team (US-CERT). US-CERT Vulnerability Note Field Descriptions. 2006 [cited 16 March 2007]. Available from URL: http://www.kb.cert.org/vuls/html/fieldhelp.

[4] SANS Institute. SANS Critical Vulnerability Analysis Archive. Undated [cited 16 March 2007]. Available from URL: http://www.sans.org/newsletters/cva/.