CVSS-SIG Version 2 History

Peter Mell, Karen Scarfone

National Institute of Standards and Technology

Gavin Reid

Cisco Systems, Inc.

Table of Contents

Introduction

This document will attempt to interpret the history and rationale behind changes made in the Common Vulnerability Scoring System (CVSS) from version 1 to version 2 (referred to as CVSS v1 and v2 in this document.) This document contains multiple appendices that provide a historic record of the stages and subsequent rationale that led to the developed standard; however, it is not intended as a CVSS user guide. For detailed how-to information, please see the complete CVSS Guide.

The Forum of Incident Response and Security Teams (FIRST) sponsors and supports the Common Vulnerability Scoring System-Special Interest Group (CVSS-SIG), a diverse group of security professionals that has a keen interest in security vulnerabilities and use CVSS in its daily work. Many group members have implemented the scoring system for varying production uses within their organizations. Their experiences have helped shape the proposed direction for CVSS, while providing continued leadership for support and training. A list of the CVSS-SIG organizations is available at http://www.first.org/cvss/eadopters.html. In addition, a list of active CVSS-SIG members is available at http://www.first.org/cvss/team/.

CVSS

The initial CVSS v1 design was not subjected to mass peer review across multiple organizations or industries. After production use, feedback from CVSS-SIG and others indicated there were significant issues with the initial draft of CVSS. To help address these, and increase the accuracy of CVSS, the CVSS-SIG began scoring vulnerabilities and comparing scores – examining inconsistencies and authoring amendments to fix them. When the CVSS-SIG reached consensus around a particular problem or potential enhancement, it voted to approve, discard, or send the amendment back to committee for further discussion and work. The last round of voting ended June 1, 2007. For detailed information on the changes see Appendix A or go to http://www.first.org/cvss/draft/accepted/cvss-11.html.

The lengthy research work into all the corner cases and mainstream scoring issues has generated much discussion among the various participants in the CVSS-SIG. It is important to note that the CVSS-SIG retroactively scored old vulnerabilities alongside new ones – an exercise that has proven beneficial because the actual significance of the vulnerabilities had been determined, which allowed the new v2 score to be compared against the impact of the real world vulnerabilities and the v1 score. In addition, the SIG scored all vulnerabilities in the NIST database to understand and compare against v1 to ensure the fidelity of the scoring increased and potentially to make changes where issues were found.

Conclusion

The CVSS-SIG team began meeting and working on CVSS v2 in April 2005 and will continue its effort up to the June 2007 release. The team has generated a large body of documentation including meeting minutes, draft versions, and other documents, some of which is available at http://www.first.org/cvss. For detailed information on the CVSS v2

vector definitions see Appendix B or go to the National Institute of Standards and Technology (NIST) website http://nvd.nist.gov/cvss.cfm?vectorinfov2.

This documentation and CVSS v2 represents the selfless work of the CVSS-SIG, with a special acknowledgment to Peter Mell and the NIST team for their work and guidance. NIST provided a group of statisticians that reviewed calculations to help reduce inconsistencies with the current formulas including rounding, lack of diversity of scores, multiplicative equations issues, high score scarcity problems, and many-to-one scoring issues. For example, using the current formulas the vulnerabilities tend to group together in a few places.

The CVSS-SIG has been working with the formulas attempting to get a better spread while increasing the accuracy of the scores. Testing and refining the formulas based on the painstaking work of the CVSS-SIG generated multiple potential scoring formulas. The development of the formulas, documentation of the draft attempts, and the subsequent rationale behind the changes is available in Appendices C through M. The documentation is not intended for CVSS scoring or an explanation of use; rather, it has been provided only to reveal the process whereby the new formula was created, tested, and finalized. The CVSS-SIG team developed the formulas from extensive real world testing conducted in 2006-2007, and it will announce the availability of CVSS v2 and the new features and formulas at http://www.first.org/cvss.

Appendix A: Accepted Proposals for Change to CVSS v1

This appendix documents the major changes made from CVSS v1 to CVSS v2. While developing CVSS v2, the CVSS-SIG discussed how to address the deficiencies in v1 and created a separate proposal for each major change to CVSS v1 that was desired. The CVSS-SIG then voted on each proposal. This appendix contains copies of all the proposals that were approved, and briefly references the proposal that was dropped.

Proposal 0

Proposal 0: Additional access complexity granularity to base scoring
Release Date: 01/03/06
Status: Approved by CVSS SIG

Description

Case to add additional granularity into AccessComplexity. We would add an additional “medium” field to help better describe the complexity needed to use a particular exploit.

Formula Change

Current CVSS 1.0: (case AccessComplexity of high: 0.8, low: 1.0)
Would change to the following (case AccessComplexity of high: 0.6, medium: 0.8, low: 1.0)

Documentation Change

Current CVSS 1.0:
High: Specialized access conditions exist; for example, the system is exploitable during specific windows of time (a race condition), the system is exploitable under specific circumstances (non-default configurations), or the system is exploitable with victim interaction (vulnerability exploitable only if user opens e-mail)

Low: Specialized access conditions or extenuating circumstances do not exist; the system is always exploitable

Approved Change

High: in most configurations, the attacker must already have high privileges; or, must control or spoof additional systems besides the target system (e.g. DNS); or, the attack can only be opportunistic, i.e. the attacker can not directly trigger the vulnerability; depends on social engineering methods that would be easily detected by knowledgeable people; or, the affected configuration is deemed to be very rare in practice; or, the race condition window is very narrow; or, depends on the presence of other vulnerabilities

Medium: the attacking party is limited to a group of systems or users at some level of authorization, possibly untrusted; there is a requirement for some information gathering before a successful attack can be launched; the affected functionality is not

always used; or requires a small amount of social engineering that might occasionally fool cautious users (e.g. phishing attacks that modify the status bar to show a false link, having to be on someone's "buddy" list before sending an IM exploit).

Low: 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 in one or two steps that require little skill or additional information gathering; the "race condition" is a lazy one, i.e. it is technically a race but easily winnable (as is the case with many symlink vulns).

Proposal 1

Proposal 1
Release Date: 3/31/06
Status: Approved by CVSS SIG

Move the "impact bias" metric from the base metric group to the environmental metric group. In its new form within the environmental section, it would enable end users to declare which CIA attribute is most important to them in the context of a particular vulnerability.

Rationale

1. Which CIA metric is of most importance usually depends on the end user environment and their use of the software. For example, encryption software is suggested by CVSS v1.0 as a good candidate for a "confidentiality bias." However, some organizations may prefer an "integrity bias" if the integrity of the data being passed is more important than the confidentiality. Just because you're encrypting data doesn't mean that you don't care more about integrity (or even availability for that matter). It all depends on the type of data being transmitted and thus this metric fits better within the environmental scoring.

2. It is difficult to accurately determine this metric for the vast majority of software packages and thus it unnecessarily complicates the standard.

3. It is difficult to get analysts to consistently assign this metric and thus it promotes score inconsistency between organizations.

See section 7.5 of the NIST CVSS paper for more discussion of this proposal and for details on how it affected NIST's overall scoring efforts.

Proposal 2

Proposal 2
Release Date: 4/4/06
Status: Approved by CVSS SIG

Explicitly add to the CVSS standard that vulnerabilities that give root level access should be CIA of all "complete" and vulnerabilities that give user level access to the OS or general access to an application should be CIA of "partial." Make it clear in the standard that we are not just rating what access the vulnerability directly gives the attacker but what access it also indirectly gives the attacker. Thus an integrity violation that allows one to modify an OS password file would be labeled CIA of all "complete." Also, we should mention that following this property means that in most cases integrity violations would enable one to cause availability problems and thus setting Integrity to "partial" should usually imply a setting of "partial" or "complete" for availability.

The documentation should make clear that the indirect benefits should not take into account any interaction with other vulnerabilities that are simultaneously present.

Example

A PHP application, by design, allows a special user to upload files. However, due to a vulnerability (issue 1), it allows the user to upload files with the .php extension, which are then executed from the web server. A separate issue 2 in the application allows an unauthenticated attacker to change their userID to arbitrary user IDs by means of a cookie. Due to the combination of these 2 issues, a remote attacker could execute arbitrary PHP code; but issue 1 alone requires authentication and special privileges, and issue 2 alone does not allow arbitrary code.

Rationale

This is commonly done by all who do large amounts of CVSS scoring but since it isn't in the standard, it causes confusion for newcomers and possibly causes unnecessary criticism of the standard by those not in the community

Proposal 3

Proposal 3: Rewording of Access Complexity
Release Date: 4/4/06
Status: Approved by CVSS SIG

Reword the access complexity definition to make it clear that the metric is dealing with how easy it is to execute an attack once exploit code exists as opposed to measuring how hard it is to write exploit code.

Rationale

1. It is difficult for analysts to determine how hard it is to write exploit code and thus this metric could be set unreliably. For example, the current standard implies that all race conditions result in attacks that are difficult to launch but this is not always the case.

2. From the NIST CVSS paper: "Measuring the difficulty of writing exploit code within this metric is problematic when used in conjunction with CVSS temporal scoring. In calculating the CVSS base score, setting access complexity from

"Low" to "High" drops a score by 20%. When calculating the temporal score, there is a metric called Exploitability, which has four options: unproven an exploit exists, proof of concept code available, functional exploit available, and exploit incorporated into major worm or virus. Once a functional exploit is available, it no longer matters whether or not it was difficult to write the exploit. Thus, in the temporal scoring one should ideally ignore or undo the effects of the exploit code-writing element of the access complexity metric whenever a proof of concept exploit is publicly available. Instead, the temporal exploitability metric can only further reduce the CVSS score due to the way the temporal score equation is constructed. This means that a vulnerability exploited by a major worm that was difficult to write will have a lower score in the CVSS temporal scoring than a vulnerability exploited by a major worm that was easy to write. However, when a major worm is released, the impact on computer systems has nothing to do with how much effort it took the virus writer to create the worm."

Proposal 4

Proposal: 4: Modify Target Distribution Environmental Metric
Status: Discarded by CVSS SIG

Change target distribution environmental metric to none/low/low-medium/medium-high/high from none/low/medium/high.

Proposal 5

Proposal 5: Additional Granularity for Access Vector
Release Date: 4/12/06
Status: Approved by CVSS SIG

Add a new option to the access vector metric for vulnerabilities that are accessible only over a local network and rename existing metrics appropriately. The new option is called "Local network accessible" (sometimes referred to as "adjacent" in the CVSS SIG email list discussions).

A vulnerability that is adjacently exploitable will have a lower score than a remotely exploitable vulnerability, and a higher score than a locally exploitable vulnerability.

Revised Access Vector Metric Definitions

1. Requires local access: The attacker must have physical access to the hardware the vulnerable software is running on, or a local shell account on the machine. The vulnerable software does not accept packets from the network stack. Examples of these would include peripheral attacks such as Firewire/USB DMA attacks, as well as local privilege escalations such as sudo privilege escalation.[1]

[1] Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify and describe the use of CVSS. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials, instruments, or equipment identified are necessarily the best available for the purpose.

2. Local Network accessible: The attacker must have access to the broadcast or collision domain that the vulnerable software is running on, or physical proximity to the hardware the vulnerable software is running on. Examples of local networks include: local IP subnet, Bluetooth, 802.11, and local Ethernet segment.

3. Network accessible: The vulnerable software is bound to the network stack, and the attacker does not require local network access. An example of a network accessible attack is a RPC buffer overflow.

Proposal 6

Proposal 6: Modification to Collateral Damage Potential
Release Date: 8/24/06
Status: Approved by CVSS SIG

1.3.1 Collateral Damage Potential

This metric measures the potential for a 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,

1.3.1.1 Collateral Damage Potential Scoring Evaluation

None: There is no potential for loss of life, physical asset, productivity or revenue

Low: A successful exploit of this vulnerability may result in light physical or property damage. Or, there may be slight loss of revenue or productivity to the organization.

Low-Medium: A successful exploit of this vulnerability may result in moderate physical or property damage. Or, there may be moderate loss of revenue or productivity to the organization.

Medium-High: A successful exploit of this vulnerability may result in significant physical or property damage or loss. Or, there may be significant loss of productivity or revenue.

High: A successful exploit of this vulnerability may result in catastrophic physical or property damage and loss. The range of effect may be over a wide area. Or, there may be catastrophic loss of productivity or revenue

Proposal 7

Proposal 7: Modification of Access Vector and Authentication Metrics
Release Date: 4/25/06
Status: Approved by CVSS SIG

Authentication Metric

1. Requires no authentication

2. Requires a single instance of authentication

3. Requires multiple instances of authentication

Note: The authentication metric measures what level of authentication/authorization is needed prior to launching the attack. The exact type of authentication is not being measured (e.g., we don't care for this metric whether or not an application authenticates using OS credentials or its own private scheme).

A single instance of authentication involves one requirement for an attacker to prove their identity. This metric does not gauge the strength or diversity of the authentication step (password and two-factor authentication are both a single instance of authentication), only that an attacker is asked to prove identity before an exploit may occur. Vulnerabilities that require an attacker to be logged in to a system (such as at a command-line or via a desktop session or web interface) represent a single instance of authentication.

Multiple instances of authentication are, therefore, two or more requirements for an attacker to prove identity. If an attacker must pass more than one request for authentication, such as authenticating to an operating system login as well as provide credentials to access an application hosted on that system, then multiple instances of authentication have been required. Again, note that the strength or diversity of authentication is not measured -- even if the credentials are identical for each instance of authentication, or if the second set of credentials is commonly "saved" (such as in an e-mail client).

Because authentication instances are measured as those required before launching an attack, the metric should be applied at the earliest stage from which an attacker can exploit the vulnerability. For example, if a mail server offers commands pre- and post- authentication that are vulnerable, the metric should score as No Authentication, because the attacker can launch the exploit before credentials are required; if the vulnerable commands are only available post-authentication, then it should be scored as a Single Instance. Also, if the vulnerability exists in an authentication scheme itself (e.g. PAM; default user account) or anonymous service (e.g. public FTP server), it may be necessary to score as No Authentication if the attacker can exploit the issue without supplying credentials.

Proposal 8

Proposal 8: Direct and Indirect Impact of Exploitation
Release Date: 6/16/06
Status: Approved by CVSS SIG

Our multi-organization scoring comparison effort has revealed that the scoring of vulnerabilities that potentially have an impact on secondary hosts that access exploited servers, such as cross site scripting (XSS) vulnerabilities, is the cause of a large source of CVSS scoring discrepancies between multiple IT security

organizations. For example, some analysts score XSS vulnerabilities with respect to the direct impact on the service, and some score them with respect to the indirect impact on an end user of the service.

In order to make scoring consistent and to focus scoring on the software that is directly vulnerable, the CVSS documentation should be updated to reflect that vulnerabilities should always be scored with respect to the impact on the vulnerable service. For the majority of cases CIA will be scored Confidentiality None, Integrity Partial, and Availability None

Proposal 9

Proposal 9: Assumptions for Application Privileges
Release Date: 8/24/06
Status: Approved by CVSS SIG

Our multi-organization scoring comparison effort has revealed that a major source of scoring discrepancies is different assumptions made by analysts as to the privileges under which various applications, such as Web servers and Web browsers, are run. For example, the scores for exploiting a Web server will be quite different if the Web server is assumed to run with root-level or user-level privileges.

To make scoring more consistent, the CVSS documentation should be updated to indicate that vulnerabilities should be scored based on the privileges that are most often used (sometimes referred to as “most probably”) for the application. This does not necessarily reflect the best practice for the application, especially for client applications, which are often run with root-level privileges. If it is not clear what privileges are most often used for an application, analysts should assume the default configuration.

Proposal 10

Proposal 10: Handling Multiple Exploitation Methods
Release Date: 8/24/06
Status: Approved by CVSS SIG

Our multi-organization scoring comparison effort has revealed that some scoring discrepancies are due to analysts taking different approaches to handling cases where there is more than one way to exploit a particular vulnerability. For example, a vulnerability could be exploited using a low-complexity method to gain user-level access, and exploited using a high-complexity method to gain root access.

To make scoring more consistent, the CVSS documentation should be updated to indicate that analysts should generate a score for each approach to exploitation and then assign the vulnerability the highest of the scores. If the highest score is shared by multiple approaches, then analysts should compare those approaches and select the one that is most likely to be used.

Current Text

"The authors recognize that many other metrics could be included in CVSS. They also realize that no one scoring system will fit everyone's need perfectly. The particular constituent metrics used in CVSS were identified as the best compromise between completeness, ease-of-use and accuracy. They represent the cumulative experience of the authors as well as extensive testing of real-world vulnerabilities in end-user environments."

Proposed Text

"The authors recognize that many other metrics could be included in CVSS. They also realize that no one scoring system will fit everyone's need perfectly. The particular constituent metrics used in CVSS were identified as the best compromise between completeness, ease-of-use and accuracy. They represent the cumulative experience of the authors as well as extensive testing of real-world vulnerabilities in end-user environments. It is important to note that CVSS scoring on the impact of a vulnerability that has multiple exploitation methods the scorer chooses the exploit that has the most impact – the scorer should not choose the most like or easiest to exploit but go with the "worst case scenario" for given exploit.

Proposal 11

Proposal 11: Clarification to the Approved Proposal #1 on Impact Bias
Release Date: 9/21/06
Status: Approved by CVSS SIG
Applicability: Environmental Section

Cast the impact bias within the environmental section (as required by the approved proposal #1) using the following group of three metrics:
System Confidentiality Requirement: Low, Medium, or High System Integrity Requirement: Low, Medium, or High System Availability Requirement: Low, Medium, or High

The idea is that a vulnerability can be rated with CIA impact levels (in the base section) and an affected system can be labeled with CIA requirements (in the environmental section) in such a way that they can be directly compared. Then the environmental score can accurately reflect the true impact of particular vulnerability types on particular systems. The resultant CVSS vector could then clearly show the relationships.

Example

A base CIA of Complete, None, None applied to an environment with CIA requirements of High, Low, Low would bias the environmental score to be higher because the vulnerability affects confidentiality and the system has high confidentiality requirements.

A base CIA of None, Complete, None applied to an environment with CIA requirements of High, Low, Low would bias the environmental score to be lower

because the vulnerability affects Integrity but the system is primarily concerned with confidentiality.

Proposal 12

Proposal 12
Release Date: 11/1/06
Status: Approved by CVSS SIG

The impact and difficulty sub-equations should be combined together using a simple weighting of .6 for impact and .4 for difficulty. This is roughly how the weightings were assigned in CVSS v1 (.572 and .428).

Here is how we came up with the relative weightings for the impact and difficulty metrics in CVSS v1:

* Possible range for CIA is
  (10.0 * (1.0 + 1.0 + 1.0)/3) = 10.0
  (10.0 * (0.0 + 0.0 + 0.0)/3) = 0.0
  Size of range = 10.0
* Possible range for AAA is
  (10.0 * 1.0 * 1.0 * 1.0) = 10.0
  (10.0 * 0.7 * 0.6 * 0.6) = 2.52
  Size of range = 7.48

7.48/(7.48 + 10.0) = 42.79%
10.0/(7.48 + 10.0) = 57.21%

Appendix B: CVSS v2 Vector Definitions

CVSS v2 Vector Definitions

Every application or service that uses the Common Vulnerability Scoring System (CVSS) should provide not only the CVSS score, but also a vector describing the components from which the score was calculated. This provides users of the score confidence in its correctness and provides insight into the nature of the vulnerability.

CVSS vectors always include base metrics and may contain temporal metrics. See the CVSS standard's guide for detailed descriptions of CVSS metrics and their possible values.

CVSS Base Vectors

CVSS vectors containing only base metrics take the following form:
(AV:[L,A,N]/AC:[H,M,L]/Au:[N,S,M]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C])

The letters within brackets represent possible values of a CVSS metric. Exactly one option must be chosen for each set of brackets. Letters not within brackets are mandatory and must be included in order to create a valid CVSS vector. Each letter or pair of letters is an abbreviation for a metric or metric value within CVSS. These abbreviations are defined below.

Example 1: (AV:L/AC:H/Au:N/C:N/I:P/A:C)
Example 2: (AV:A/AC:L/Au:M/C:C/I:N/A:P)

Metric: AV = AccessVector (Related exploit range)
Possible Values: L = Local access, A = Adjacent network, N = Network

Metric: AC = AccessComplexity (Required attack complexity)
Possible Values: H = High, M = Medium, L = Low

Metric: Au = Authentication (Level of authentication needed to exploit)
Possible Values: N= None required, S= Requires single instance, M= Requires multiple instances

Metric: C = ConfImpact (Confidentiality impact)
Possible Values: N = None, P = Partial, C = Complete

Metric: I = IntegImpact (Integrity impact)
Possible Values: N = None, P = Partial, C = Complete

Metric: A = AvailImpact (Availability impact)
Possible Values: N = None, P = Partial, C = Complete

CVSS Temporal Vectors

CVSS vectors containing temporal metrics are formed by appending the temporal metrics to the base vector. The temporal metrics appended to the base vector take the following form:
/E:[U,P,F,H]/RL:[O,T,W,U]/RC:[N,U,C]

Example 1: (AV:L/AC:H/Au:N/C:N/I:P/A:C/E:P/RL:O/RC:C)
Example 2: (AV:LN/AC:L/Au:M/C:C/I:N/A:P/E:F/RL:T/RC:U)

Metric: E = Exploitability (Availability of exploit)
Possible Values: U = Unproven, P = Proof-of-concept, F = Functional, W = Widespread

Metric: RL = RemediationLevel (Type of fix available)
Possible Values: O = Official-fix, T = Temporary-fix, W = Workaround, U = Unavailable

Metric: RC = ReportConfidence (Level of verification that the vulnerability exists)
Possible Values: N = Not confirmed, U = Uncorroborated, C = Confirmed

CVSS Environmental Vectors

CVSS vectors containing environmental metrics are formed by appending the environmental metrics to the temporal vector. The environmental metrics appended to the temporal vector take the following form:
/CD[N,L,LM,MH,H]:/TD:[N,L,M,H]/CR:[L,M,H]/IR:[L,M,H]/AR:[L,M,H]

Example 1:
(AV:L/AC:H/Au:N/C:N/I:P/A:C/E:P/RL:O/RC:C/CD:L/TD:M/CR:L/IR:L/AR:H)
Example 2:
(AV:LN/AC:L/Au:M/C:C/I:N/A:P/E:F/RL:T/RC:U/CD:MH/TD:H/CR:M/IR:L/AR:M)

Metric: CD = Collateral Damage Potential (Organization specific potential for loss)
Possible Values: N = None, L = Low, LM = Low-Medium, MH = Medium-High, H = High

Metric: TD = Target Distribution (Percentage of vulnerable systems)
Possible Values: N = None (0%), L = Low (1-25%), M = Medium (26-75%), H = High (76-100%)

Metric: CR = System Confidentiality Requirement (draft proposal)
Possible Values: L = Low, M = Medium, H = High

Metric: IR = System Integrity Requirement (draft proposal)
Possible Values: L = Low, M = Medium, H = High

Metric: AR = System Availability Requirement (draft proposal)
Possible Values: L = Low, M = Medium, H = High

CVSS Vectors and CVSS Compatible Products

CVSS compatible products may provide their users access to the NVD CVSS v2 calculator by creating a hyperlink that includes the CVSS vector and, optionally, the vulnerability name. This works for both base and temporal vectors. The hyperlinks should take one of the following forms.

Example base vector hyperlinks to CVSS calculator (with and without vulnerability name):

- http://nvd.nist.gov/cvss.cfm?version=2&vector=(AV:L/AC:H/Au:N/C:N/I:P/A:C)
- http://nvd.nist.gov/cvss.cfm?version=2&name=example&vector=(AV:A/AC:L/Au:M/C:C/I:N/A:P)

Example temporal vector hyperlinks to CVSS calculator (with and without vulnerability name):

- http://nvd.nist.gov/cvss.cfm?version=2&vector=(AV:L/AC:H/Au:N/C:N/I:P/A:C/E:P/RL:O/RC:C)
- http://nvd.nist.gov/cvss.cfm?version=2&name=example&vector=(AV:A/AC:L/Au:M/C:C/I:N/A:P/E:F/RL:T/RC:U)

Appendix C: Impact and Difficulty Sub-equation Ordering Scoring Rules

In CVSS v1, the equation for the base score was composed of several metrics. In CVSS v2, these metrics were divided into two groups – impact and difficulty – and the base equation was split into two sub-equations, one for each metric group. For each sub-equation, the CVSS-SIG analyzed the relative severity of each possible combination of metrics and developed a set of verbal rules that described the results of the analysis, such as vectors with a certain combination of metrics being scored higher than vectors with another combination of metrics. This appendix lists the verbal rules for each sub-equation.

Impact Sub-Equation Ordering/Scoring Rules

Impact Sub-Equation Vector: (C:[N,P,C]/I:[N,P,C]/A:[N,P,C])
CVSS v2 vector information: http://nvd.nist.gov/cvss.cfm?vectorinfov2

Rule 1: Unless this rule is superseded by another rule, the CIA ordering should follow the ordering implicit within CVSS v1.

Rule 1 Rationale: CVSS v1 has worked well in general and so the implicit orderings should be followed except within specific areas we target for improvement.

Rule 2: CIA vectors with one "Complete" component must be rated higher than those without any "Complete" components (except for the vector that has 3 "Partial" components).

Rule 2 Rationale: The purpose of this rule is to raise the importance of "Complete" components as compared to "Partial" components. This rule is necessary based on Steve Christey's analysis of CVE vulnerabilities where he discovered that vulnerabilities with a single "Complete" component were of significant severity qualitatively but the CVSS v1 score was very low (e.g., 2.3). The reason to exclude the vector that has 3 "Partial" components from this rule is that (C:P/I:P/A:P) represents a significant incursion (obtaining user level OS access or admin application access) and we don't want to force this vector to a lower score than it has in CVSS v1.

Rule 3: Confidentiality and Integrity violations have somewhat more impact than availability violations.

Rule 3 Rationale: This was highlighted as a need in the NIST CVSS paper (see excerpt below). The CVSS SIG discussed this and the consensus appeared to be that "C" and "I" should be given more weight than "A." Note that the NIST CVSS paper also argued that "I" should be given more weight than "C" but we never achieved consensus on that within the SIG.

"Integrity should be given more weight than availability because violations of integrity allow violations of availability. Alternately, it could be required that availability always be set at least as high as integrity. If an attacker can make arbitrary changes to a system, then changes could be made to negatively impact availability. For example, if one can delete the data (an integrity violation) then the data can be made unavailable (an availability violation). Additionally, exploitation of integrity has greater impact because it is often difficult to notice the violation, difficult to determine what was changed, and difficult to recover the data to a clean state. This is not true with exploitation of availability.

Confidentiality should be given more weight than availability because many violations of confidentiality are non-recoverable (once the data is stolen it cannot be taken back), while violations of availability are usually easily recoverable. Also, the exploitation of availability is typically noticed very quickly, but violations of confidentiality are hard to detect."

Rule 4: Vectors with 2 "Complete" components must be rated higher than those with no "Complete" components

Rule 4 Rationale: A vector with 2 "Complete"components almost has complete control over a computer and thus should be rated higher than even a vector with 3 "Partials."

Rule 5: (C:C/I:C/A:C) and (C:P/I:P/A:P) vulnerabilities should be rated closer together than in CVSS v1.

Rule 5 Rationale: (C:C/I:C/A:C) and (C:P/I:P/A:P) vectors represent one gaining an admin OS account and a user level OS account respectively. It is often difficult or even impossible for an analyst to distinguish which level of access is provided by a vulnerability because it often depends on how the vulnerable application was installed (e.g., one can optionally install software as root/admin or as a user). Thus, both vectors should score high and be fairly close to each other (e.g., separated by no more than 2).

NIST CVSS paper in discussion of sources of error: "One source of error is the decision where an analyst must decide whether a vulnerability gives root level access to an operating system or user level access. This is not always clear from the advisory and it may depend on how a system administrator installs the software. This ambiguity may have increased the significance of the spike at score 7, as the analysts appear to have avoided claiming that a vulnerability gives root access unless it is clear that this is the case."

Difficulty Sub-Equation Ordering/Scoring Rules

Difficulty Sub-Equation Vector: (AV:[N,LN,L]/AC:[L,M,H]/Au:[N,S,M])
CVSS v2 vector information: http://nvd.nist.gov/cvss.cfm?vectorinfov2

Rule 1: Unless this rule is superseded by another rule, the AAA ordering should follow the ordering implicit within CVSS v1.

Rule 1 rationale: CVSS v1 has worked well in general and so the implicit orderings should be followed except within specific areas we target for improvement.

Rule 2: If two vectors have the same AV and Au metrics, the relative difference in difficulty is greater when comparing AC:M to AC:H than it is when comparing AC:L to AC:M.

Rule 2 rationale: AC:L allows attacks to be performed essentially at will (ignoring all other metrics). AC:M also allows attacks to be performed fairly easily, but against fewer targets or with greater risk of detection, so AC:M is a minor increase in difficulty from a vulnerability with AC:L. AC:H vulnerabilities are generally very difficult to exploit, making them generally unattractive targets, so AC:H is a major increase in difficulty from a vulnerability with AC:M. In CVSS v1, going from AC:L to AC:M caused a 20% drop in score, and going from AC:M to AC:H caused a 25% drop in score. We assert that the difference should be greater than 20% versus 25%.

Rule 3: If two vectors have the same AV and AC metrics, the relative difference in difficulty is roughly equal when comparing Au:N to Au:S and when comparing Au:S to Au:M.

Rule 3 rationale: Au:N allows attacks to be performed essentially at will (ignoring all other metrics). Au:S requires attackers to guess, sniff, or otherwise determine a password or other authentication mechanism (most often, a password). Au:M requires attackers to guess two or more authentication mechanisms (most often, passwords), which in some case will be identical (e.g., same password for OS and application level access). So in some cases, Au:M will be much more difficult than Au:S, while in other cases Au:M will not be substantially more difficult (simply authenticating a second time using the same credentials). In CVSS v1, going from Au:N to Au:S caused a 20% drop in score, and going from Au:S to Au:M caused a 25% drop in score. We assert that the drops should be roughly equal.

Rule 4: The presence of the metric AC:H in a vulnerability should cause a greater reduction in the score than the presence of the metric Au:M.

Rule 4 rationale: A vulnerability with AC:H is always extremely difficult to exploit. A vulnerability with Au:M is sometimes extremely difficult to exploit (e.g., determining multiple types of authentication credentials), and sometimes relatively easy to exploit (e.g., guessing or sniffing one password and using it for two sets of authentications). Therefore, the presence of Au:M should not reduce a score as much as the presence of AC:H. In CVSS v1, AC:H and Au:M each caused a 40% drop in score.

Rule 5: The presence of the metric Au:S in a vulnerability should cause a somewhat greater reduction in the score than the presence of the metric AC:M.

Rule 5 rationale: A vulnerability with Au:S requires attackers to guess, sniff, or otherwise determine a password or other authentication mechanism (most often, a password). A vulnerability with AC:M generally allows attacks to be performed fairly easily at will. We assert that on average it is somewhat more likely for an attacker to succeed against AC:M compared to Au:S. In CVSS v1, AC:M and Au:S each caused a 20% drop in score.

Rules 2 through 5 collectively rank the AC and Au metrics as follows:

AC:L, Au:N (highest score)
AC:M
Au:S
Au:M
AC:H (lowest score)

There are gaps of varying sizes between the metrics. For example, the change from Au:N to Au:S (the first to the third line) should have roughly the same reduction in score as the change from Au:S to Au:M (the third to the fourth line).

Rule 6: The AV metrics are of relatively little importance compared to the AC or Au metrics. AV:L and AV:LN are of less importance than AC:M or Au:S.

Rule 6 rationale: Many current attacks (particularly malware-based ones) exploit vulnerabilities that are AV:L or AV:LN. Unless a host is directly accessible via the Internet (which is not true for most hosts), AV:N is really limited to the same pool of attackers as AV:LN or AV:L. In most cases, AV:N, AV:LN and AV:L are not very much different in terms of attack difficulty. Therefore, the AV metrics should have little impact on scoring as compared to the AC or AU metrics. In CVSS v1, the AV metrics have less impact on scoring (a maximum of 30% reduction in score for AV, compared to 40% for AC or Au). We assert that the AV metrics should have much less impact on scoring (less impact than AC:M or Au:S).

Appendix D: CVSS 2.0 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the first draft of the CVSS v2 base score equation.

CVSS v2 Base Score Equation

Base Score Equation =
(.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication-AccessVector*10+.06))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

AccessVector = case AccessVector of
Requires local access: 0.04
Local Network accessible: 0.02
Network accessible: 0

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

Authentication = case Authentication of
Requires no authentication: 0.7
Requires single instance of authentication: 0.56
Requires multiple instances of authentication: 0.45

Correspondence

The following is correspondence from James H. Yen, Statistical Engineering Division, NIST:

Following up my previous email, I have tweaked my equation to try to achieve better separation between adjacent scores and to have CCC have a perfect (storm) 10 score.

To that end, the model depends on 3 parameters called x.3, x.7, and x10 (so named because in the original equation I had x.3= .3, x.7 = .7, and x10 =10), although x10 will be a function of x.7, so there are really only two parameters.

The modified fit goes as follows, using similar notation as before:

CC(N)=II(N)=AA(N) = 0
CC(P)=II(P)=AA(P) = x.3
CC(C)=II(C)=AA(C) = x.7

Let the fitting function be:

V2 = x10 * [1 - (1-CC)*(1-II)*(1-AA)].

There is probably a way to optimize the problem numerically, but doing trial and error gives one plausible set of parameters:

x.3 = .275
x.7 = .66
x10 =1.041

Note that x10 is there to normalize the CCC score to be 10.0.

These lead to the fit below, which satisfies all criterion except that the scores of 9.21 and 9.54 are still too close together. I can adjust x.3 and x.7 to get a better separation of those but that leads to the scores for PPP and CNN/NCN/NNC being too close, and since those scores are not “rare,” considered that fit to be worse than the one listed here.

C	I	A	cvs2	Fit
C	C	C	10	10
C	C	P	9.5	9.54
C	P	C	9.5	9.54
P	C	C	9.5	9.54
C	C	N	9	9.21
C	N	C	9	9.21
N	C	C	9	9.21
C	P	P	8.5	8.55

P	C	P	8.5	8.55
P	P	C	8.5	8.55
P	P	P	6.3-6.9	6.44
C	P	N	8	7.84
C	N	P	8	7.84
P	C	N	8	7.84
N	C	P	8	7.84
P	N	C	8	7.84
N	P	C	8	7.84
C	N	N	7	6.87
N	C	N	7	6.87
N	N	C	7	6.87
P	P	N	5	4.94
P	N	P	5	4.94
N	P	P	5	4.94
P	N	N	3	2.86
N	P	N	3	2.86
N	N	P	3	2.86
N	N	N	0	0

Appendix E: CVSS v2.1 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the second draft of the CVSS v2 base score equation. The only change from the first draft is in the values assigned to the AccessVector metric.

CVSS v2.1 Base Score Equation

Base Score Equation =
(.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication-AccessVector*10+.06))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

AccessVector = case AccessVector of
Requires local access: 0.3
Local Network accessible: 0.15
Network accessible: 0

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

Authentication = case Authentication of
Requires no authentication: 0.7
Requires single instance of authentication: 0.56
Requires multiple instances of authentication: 0.45

Appendix F: CVSS v2.2 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the third draft of the CVSS v2 base score equation. The changes from the previous draft are a minor adjustment to the equation itself and changes to the values assigned to the AccessVector, AccessComplexity, and Authentication metrics.

CVSS v2.2 Base Score Equation

Base Score Equation = (.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) + .4*(20*AccessComplexity*Authentication-AccessVector*10))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

AccessVector = case AccessVector of
Requires local access: 0.375
Local Network accessible: 0.188
Network accessible: 0

AccessComplexity = case AccessComplexity of
high: 0.382
medium: 0.61
low: 0.71

Authentication = case Authentication of
Requires no authentication: 0.704
Requires single instance of authentication: 0.570
Requires multiple instances of authentication: 0.491

Appendix G: CVSS v2.3 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the fourth draft of the CVSS v2 base score equation. The only change from the previous draft is a minor adjustment to the base score equation.

CVSS v2.3 Base Score Equation

Base Score Equation =
(.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication-10*AccessVector))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

AccessVector = case AccessVector of
Requires local access: 0.375
Local Network accessible: 0.188
Network accessible: 0

AccessComplexity = case AccessComplexity of
high: 0.382
medium: 0.61
low: 0.71

Authentication = case Authentication of
Requires no authentication: 0.704
Requires single instance of authentication: 0.570
Requires multiple instances of authentication: 0.491

Appendix H: CVSS v2.5 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains what is believed to be the fifth draft of the CVSS v2 base score equation. No record of CVSS v2.4 has been found, so it appears that a numbering error may have been made when CVSS v2.5 was released. The changes from the previous draft are modifications to the base score equation calculations and changes to the values of the AccessComplexity, Authentication, and AccessVector metric.

CVSS v2.5 Base Score Equation

Base Score Equation =
(.5*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.5*(20*AccessComplexity*Authentication*AccessVector))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

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

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

AccessVector = case AccessVector of
Requires local access: 0.513
Local Network accessible: 0.757
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

```
AvailImpact  = case AvailabilityImpact of
               none:        0
               partial:     0.275
               complete:    0.660
```

Appendix I: CVSS v2.6 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the sixth draft of the CVSS v2 base score equation. The changes from the previous draft are modifications to the base score equation calculations and changes to the values of the AccessVector metric.

CVSS v2.6 Base Score Equation

Base Score Equation =
(.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication*AccessVector))*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1 otherwise

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

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

AccessVector = case AccessVector of
Requires local access: .470
Local Network accessible: .735
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

Appendix J: CVSS v2.7 Base Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the seventh draft of the CVSS v2 base score equation. The changes from the previous draft are modifications to the base score equation calculations and changes to the values of the AccessVector metric. This appendix also contains an interim draft based on the v2.7 base score equation. It is identical to the draft presented in the first half of this appendix, except for changing one of the values assigned to the AccessVector metric.

CVSS v2.7 Base Score Equation

Base Score Equation =
((.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

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

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

AccessVector = case AccessVector of
Requires local access: .470
Local Network accessible: .646
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

CVSS v2.7a Base Score Equation

Base Score Equation =
((.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

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

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

AccessVector = case AccessVector of
Requires local access: .395 (this was .450 in version 2.7)
Local Network accessible: .646
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

Appendix K: CVSS v2.8 Environmental Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the first draft of the CVSS v2 environmental score equation.

CVSS Environmental Metric Formula v2.0

Environmental Equation =
((.6*(Max(10.41*(1-(1-ConfImpact*ConfReq)*(1-IntegImpact*IntegReq)*(1-AvailImpact*AvailReq))),10) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)
* Exploitability * RemediationLevel * ReportConfidence

* TargetDistribution

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

EnvironmentalScore = round_to_1_decimal(
((TemporalScore + ((10 - TemporalScore) * CollateralDamagePotential)) *
TargetDistribution))

CollateralDamagePotential = case CollateralDamagePotential of
none: 0
low: 0.1
medium: 0.3
high: 0.5

TargetDistribution = case TargetDistribution of
none: 0
low: 0.25
medium: 0.75
high: 1.00

ConfReq = case ConfidentialityImpact of
Low: 0
Medium: 0.5
High: 1.5

IntegReq = case IntegrityImpact of
Low: 0
Medium: 0.5
High: 1.5

AvailReq = case AvailabilityImpact of
Low: 0
Medium: 0.5
High: 1.5

Appendix L: CVSS v2.8 Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the eighth draft of the CVSS v2 base score equation, the first draft of the CVSS v2 temporal equation, and the second draft of the CVSS v2 environmental score equation. The changes from the previous drafts are the creation of the impact and exploitability sub-equations within the base score equation, and the revision of the environmental score equation (including a reorganization and renaming of its components).

CVSS v2.8 Equations

CVSS Base Score Equation

Base Score =
((.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

Impact Subequation = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))*f(x)/1.176

Exploitability Subequation = 20*AccessComplexity*Authentication*AccessVector

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

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

AccessVector = case AccessVector of
Requires local access: .395 (this was .450 in version 2.7)
Local Network accessible: .646
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

CVSS Temporal Equation

TemporalScore = BaseScore * Exploitability * RemediationLevel * ReportConfidence

Exploitability = case Exploitability of
unproven: 0.85
proof-of-concept: 0.9
functional: 0.95
high: 1.00

RemediationLevel = case RemediationLevel of
official-fix: 0.87
temporary-fix: 0.90
workaround: 0.95
unavailable: 1.00

ReportConfidence = case ReportConfidence of
unconfirmed: 0.90
uncorroborated: 0.95
confirmed: 1.00

CVSS Environmental Equation

Environmental Equation =
(Impact_Biased_Temporal + ((10 - Impact_Biased_Temporal)
* CollateralDamagePotential)) * TargetDistribution

Impact_Biased_Temporal = ((.6*(Min(10.41*(1-(1-ConfImpact*ConfReq)*(1-IntegImpact*IntegReq)*(1-AvailImpact*AvailReq))),10) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)
* Exploitability * RemediationLevel * ReportConfidence

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

```
CollateralDamagePotential  = case CollateralDamagePotential of
                none:     0
                low:      0.1
                medium:   0.3
                high:     0.5

TargetDistribution  = case TargetDistribution of
                none:     0
                low:      0.25
                medium:   0.75
                high:     1.00

ConfReq  = case ConfidentialityImpact of
                Low:      0
                Medium:   0.5
                High:     1.5

IntegReq  = case IntegrityImpact of
                Low:      0
                Medium:   0.5
                High:     1.5

AvailReq  = case AvailabilityImpact of
                Low:      0
                Medium:   0.5
                High:     1.5
```

Appendix M: CVSS v2.9 Score Equation

During the development of CVSS v2, several versions of the equations and metric values were developed, reviewed, and analyzed by the CVSS-SIG. Appendices D through M document the work-in-progress drafts of the equations and metric values in chronological order. This appendix contains the third draft of the CVSS v2 environmental score equation (the base and temporal score equations are unchanged from the previous draft). The change to the environmental equation is adding another possible value to the CollateralDamagePotential metric.

CVSS v2.9 Equations

CVSS Base Score Equation

Base Score =
((.6*(10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

Impact Subequation = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))*f(x)/1.176

Exploitability Subequation = 20*AccessComplexity*Authentication*AccessVector

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

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

AccessVector = case AccessVector of
Requires local access: .395 (this was .450 in version 2.7)
Local Network accessible: .646
Network accessible: 1

ConfImpact = case ConfidentialityImpact of
none: 0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0
partial: 0.275
complete: 0.660

CVSS Temporal Equation

TemporalScore = BaseScore * Exploitability * RemediationLevel * ReportConfidence

Exploitability = case Exploitability of
unproven: 0.85
proof-of-concept: 0.9
functional: 0.95
high: 1.00

RemediationLevel = case RemediationLevel of
official-fix: 0.87
temporary-fix: 0.90
workaround: 0.95
unavailable: 1.00

ReportConfidence = case ReportConfidence of
unconfirmed: 0.90
uncorroborated: 0.95
confirmed: 1.00

CVSS Environmental Equation

Environmental Equation =
(Impact_Biased_Temporal + ((10 - Impact_Biased_Temporal) * CollateralDamagePotential)) * TargetDistribution

Impact_Biased_Temporal = ((.6*(Min((10.41*(1-(1-ConfImpact*ConfReq)*(1-IntegImpact*IntegReq)*(1-AvailImpact*AvailReq))),10) +
.4*(20*AccessComplexity*Authentication*AccessVector))-1.5)*f(x)
* Exploitability * RemediationLevel * ReportConfidence

f(x)= 0 if ConfImpact=IntegImpact=AvailImpact=0
1.176 otherwise

CollateralDamagePotential = case CollateralDamagePotential of
none: 0
low: 0.1
low-medium: 0.3
medium-high: 0.4
high: 0.5

TargetDistribution = case TargetDistribution of
none: 0
low: 0.25
medium: 0.75
high: 1.00

ConfReq = case ConfidentialityImpact of
Low: 0
Medium: 0.5
High: 1.5

IntegReq = case IntegrityImpact of
Low: 0
Medium: 0.5
High: 1.5

AvailReq = case AvailabilityImpact of
Low: 0
Medium: 0.5
High: 1.5

Appendix N: Acknowledgments

The CVSS-SIG acknowledges the CVSS pioneers, who wrote the original NIAC/CVSS document:

Dave Ahmad
Symantec

Gerhard Eschelbeck
Qualys

Sasha Romanosky
Carnegie Mellon University

Mike Schiffman
Cisco Systems

Andrew Wright
Cisco Systems

The entire CVSS-SIG group has helped in the creation of the revised standard and the following are credited for continued effort in bringing the standard forward. The authors apologize for any inadvertent omissions.

Peter Allor
ISS

Mike Caudill
Cisco Systems

Steven Christey
MITRE

Anton Chuvakin
LogLogic

Gerhard Eschelbeck
Webroot

Seth Hanford
Cisco Systems

Luann Johnson
ISS

Tim “TK” Keanini, CTO
nCircle Network Security

Art Manion
US-CERT

Peter Mell
NIST

Gavin Reid
CSIRT Cisco Systems

Sasha Romanosky
Carnegie Mellon University

Karen Scarfone
NIST

Michael Scheck
CSIRT Cisco Systems

Robin Sterzer
CSIRT Cisco Systems

James H. Yen
Statistical Engineering Division, NIST