

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1525

(04/2015)

SERIES X: DATA NETWORKS, OPEN SYSTEM
COMMUNICATIONS AND SECURITY

Cybersecurity information exchange – Vulnerability/state
exchange

Common weakness scoring system

Recommendation ITU-T X.1525

ITU-T



ITU-T X-SERIES RECOMMENDATIONS
DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

PUBLIC DATA NETWORKS	X.1–X.199
OPEN SYSTEMS INTERCONNECTION	X.200–X.299
INTERWORKING BETWEEN NETWORKS	X.300–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	X.600–X.699
OSI MANAGEMENT	X.700–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	X.850–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999
INFORMATION AND NETWORK SECURITY	
General security aspects	X.1000–X.1029
Network security	X.1030–X.1049
Security management	X.1050–X.1069
Telebiometrics	X.1080–X.1099
SECURE APPLICATIONS AND SERVICES	
Multicast security	X.1100–X.1109
Home network security	X.1110–X.1119
Mobile security	X.1120–X.1139
Web security	X.1140–X.1149
Security protocols	X.1150–X.1159
Peer-to-peer security	X.1160–X.1169
Networked ID security	X.1170–X.1179
IPTV security	X.1180–X.1199
CYBERSPACE SECURITY	
Cybersecurity	X.1200–X.1229
Countering spam	X.1230–X.1249
Identity management	X.1250–X.1279
SECURE APPLICATIONS AND SERVICES	
Emergency communications	X.1300–X.1309
Ubiquitous sensor network security	X.1310–X.1339
PKI related Recommendations	X.1340–X.1349
CYBERSECURITY INFORMATION EXCHANGE	
Overview of cybersecurity	X.1500–X.1519
Vulnerability/state exchange	X.1520–X.1539
Event/incident/heuristics exchange	X.1540–X.1549
Exchange of policies	X.1550–X.1559
Heuristics and information request	X.1560–X.1569
Identification and discovery	X.1570–X.1579
Assured exchange	X.1580–X.1589
CLOUD COMPUTING SECURITY	
Overview of cloud computing security	X.1600–X.1601
Cloud computing security design	X.1602–X.1639
Cloud computing security best practices and guidelines	X.1640–X.1659
Cloud computing security implementation	X.1660–X.1679
Other cloud computing security	X.1680–X.1699

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T X.1525

Common weakness scoring system

Summary

Recommendation ITU-T X.1525 on the common weakness scoring system (CWSS) provides an open framework for communicating the characteristics and impact of information and communication technologies (ICT) weaknesses during development of software capabilities. The goal of this Recommendation is to enable ICT software developers, managers, testers, security vendors and service suppliers, buyers, application vendors and researchers to speak from a common language of scoring ICT weaknesses that could manifest as vulnerabilities when the software is used.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T X.1525	2015-04-17	17	11.1002/1000/12357

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2015

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

	Page
1 Scope.....	1
2 References.....	1
3 Definitions	1
3.1 Terms defined elsewhere	1
3.2 Terms defined in this Recommendation.....	2
4 Abbreviations and acronyms	2
5 Conventions	3
6 Use of CWSS.....	3
6.1 CWSS description	3
6.2 Workings of CWSS	4
6.3 CWSS scoring	5
6.4 CWSS users	6
7 Metric groups.....	7
7.1 Metric group factors	7
7.2 Values for uncertainty and flexibility.....	8
7.3 Base finding metric group	9
7.4 Attack Surface metric group.....	15
7.5 Environmental Metric group	19
7.6 CWSS score formula	24
7.7 CWSS vectors, scoring examples and score portability.....	26
Bibliography.....	30

Introduction

Software developers often face hundreds or thousands of individual bug reports for weaknesses that are discovered in their code. In certain circumstances, a software weakness can even lead to an exploitable vulnerability. Due to this high volume of reported weaknesses, stakeholders are often forced to prioritize which issues they should investigate and fix first. In short, people need to be able to reason and communicate about the relative importance of different weaknesses. While various scoring methods are used today, they are either ad hoc or inappropriate for application to the still-imprecise evaluation of software security. The common weakness scoring system (CWSS) provides a mechanism for prioritizing software weaknesses in a consistent, flexible, open manner while accommodating context for the various business domains and intended uses of the software. It is a collaborative, community-based effort that is addressing the needs of its stakeholders across government, academia and industry.

ICT software developers, managers, testers, security vendors and service suppliers, buyers, application vendors and researchers must identify and assess weaknesses in software that could manifest as vulnerabilities when the software is used. They then need to be able to prioritize these weaknesses and determine which to remediate based on which of them pose the greatest risk. When there are so many to fix, with each being scored using different scales, the various ICT community members, managers, testers, buyers and developers are left to their own methodologies to find some way of comparing disparate weaknesses and translating them into actionable information.

Because CWSS standardizes the approach for characterizing weaknesses, users of CWSS can invoke attack surface and environmental metrics to apply contextual information that more accurately reflects the risk to the software capability given the unique business context it will function within and the unique business capability it is meant to provide. This allows them to make more informed decisions when trying to mitigate risks posed by weaknesses.

CWSS leverages existing work from within the cyber security community such as the large number of diverse real-world publicly known vulnerabilities specified through [b-ITU-T X.1520] – common vulnerabilities and exposures (CVE) and the scoring system used for discussing the severity of those publicly known vulnerabilities through [b-ITU-T X.1521] – common vulnerability scoring system (CVSS) as well as the common weakness enumeration (CWE) list of weaknesses in the software's architecture, design, code or deployment. In constructing CWSS, the ability to allow reasonable default values for areas that may not be known, while providing for the tailoring based on business and technical context.

CWSS is one of a class of ITU-T Recommendations that comes from a large, existing, global development and user community that has written and evolved an open specification that is made available to the ITU-T for adoption with agreement that any changes or updates to the specification will be done in a manner that ensures full technical equivalency and compatibility will be maintained, that discussions about changes and enhancements will be done through the original user community processes, and includes explicit reference to the corresponding specific version maintained by the user community. Thus, at the time of initial adoption of Recommendation ITU-T X.1525, a due diligence verification and statement of equivalency will occur; and as changes are effected among the user community, timely reflection of those changes will be reflected in subsequent versions of the Recommendation through continued collaboration.

Recommendation ITU-T X.1525 - Common weakness scoring system (CWSS) has been developed on a collaborative basis with The MITRE Corporation bearing in mind the importance of maintaining, to the extent possible, technical compatibility between Recommendation ITU-T X.1525 – Common weakness scoring system (CWSS) and the "Common Weakness Scoring System (CWSS)", version 1.0.1, dated 5 September 2014 [https://cwe.mitre.org/cwss/cwss_v1.0.1.html].

Recommendation ITU-T X.1525

Common weakness scoring system

1 Scope

This Recommendation provides a standardized approach for communicating the characteristics and impacts of weaknesses during development of ICT software capabilities using attack surface and environmental metrics to apply contextual information. CWSS more accurately reflects the risk to the user of the software capability, given the unique business context it will function within for the user, and the unique business capability the software is providing to the user.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

None.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 access [b-ITU-T X.1521]: A subject's ability to view, modify, or communicate with an object. Access enables the flow of information between the subject and the object.

3.1.2 availability [b-ITU-T X.1521]: The reliability and timely access to data and resources by authorized individuals.

3.1.3 attack instance [b-ITU-T X.1544]: A specific detailed attack against an application or system targeting vulnerabilities or weaknesses in that system.

3.1.4 confidentiality [b-ITU-T X.1521]: A security principle that works to ensure that information is not disclosed to unauthorized subjects.

3.1.5 integrity [b-ITU-T X.1521]: A security principle that makes sure that information and systems are not modified maliciously or accidentally.

3.1.6 risk [b-ITU-T X.1521]: The relative impact that an exploited vulnerability would have to a user's environment.

3.1.7 threat [b-ITU-T X.1521]: The likelihood or frequency of a harmful event occurring.

3.1.8 vulnerability [b-ITU-T X.1500]: Any weakness that could be exploited to violate a system or the information it contains.

3.1.9 weakness [b-ITU-T X.1524]: A shortcoming or imperfection in the software code, design, architecture, or deployment that, could, at some point become a vulnerability, or could contribute to the introduction of other vulnerabilities.

3.2 Terms defined in this Recommendation

This Recommendation defines the following term:

3.2.1 vignette: A vignette provides a shareable, formalized way to define a particular environment, the role that software plays within that environment and an organization's priorities with respect to software security.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AI	Authentication Instances
AL	Acquired privilege Layer
AP	Acquired Privilege
AS	Authentication Strength
ASLR	Address Space Layout Randomization
AV	Access Vector
BI	Business Impact
BVC	Business Value Context
CD	Compact Disc
CIO	Chief Information Officer
CSO	Chief Security Officer
CSRF	Cross-Site-Request-Forgery
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
DI	Likelihood of Discovery
DNS	Domain Name System
DS	Deployment Scope
EC	External Control effectiveness
EX	Likelihood of Exploit
FC	Finding Confidence
FTP	File Transfer Protocol
HTML	Hyper Text Markup Language
IC	Internal Control Effectiveness
ICT	Information Communication Technology
IN	level of Interaction
IP	Internet Protocol
NIST	National Institute of Standards and Technology
OS	Operating System

OWASP	Open Web Application Security Project
P	Prevalence
PCI DSS	Payment Card Industry Data Security Standard
RL	Required Privilege Layer
RP	Required Privilege
SAMATE	Software Assurance Metrics And Tool Evaluation
SANS	SysAdmin, Audit, Networking and Security
SQL	Structured Query Language
SSL	Secure Sockets Layer
TI	Technical Impact
TLS	Transport Layer Security
USB	Universal Serial Bus
XSS	Cross-Site Scripting

5 Conventions

None.

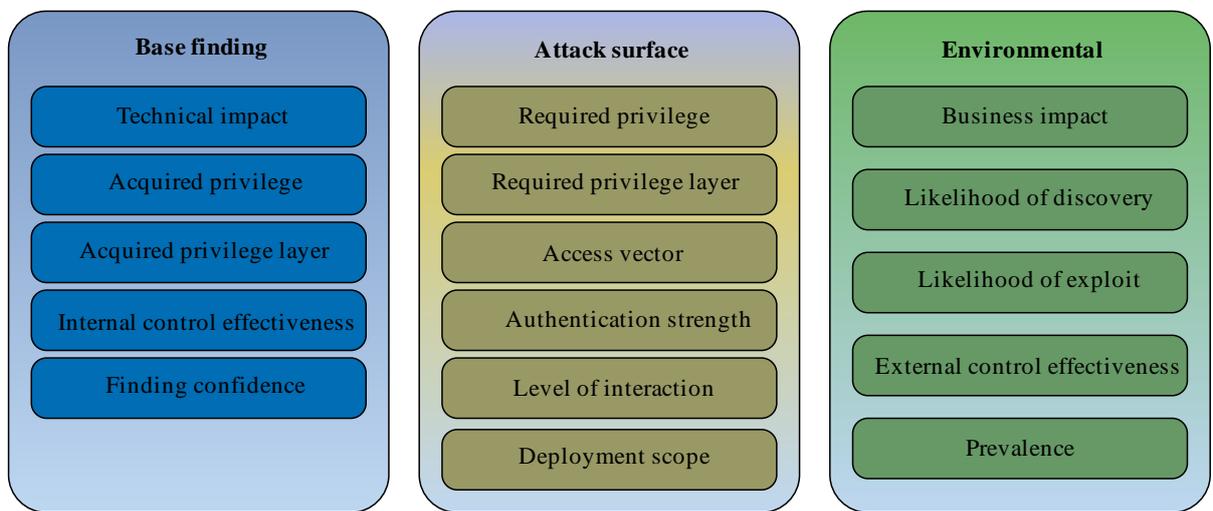
6 Use of CWSS

Currently, ICT software developers, managers, testers, security vendors and service suppliers, buyers, application vendors and researchers must identify and assess weaknesses in software that could manifest as vulnerabilities when the software is used. They then need to be able to prioritize these weaknesses and determine which to remediate based on which of them pose the greatest risk. When there are so many to fix, with each being scored using different scales, the various ICT community members, managers, testers, buyers and developers are left to their own methodologies to find some way of comparing disparate weaknesses and translating them into actionable information. The common weakness scoring system (CWSS) is an open framework that addresses this issue. It offers the following benefits:

- Quantitative measurements: CWSS provides a quantitative measurement of the unfixed weaknesses that might be present within a software application.
- Common framework: CWSS provides a common framework for prioritizing security errors ("weaknesses") that are discovered in software applications.
- Customized prioritization: in conjunction with the [Common Weakness Risk Analysis Framework \(CWRAF\) \[b-CWRAF\]](#), CWSS can be used by consumers to identify the most important types of weaknesses for their business domains, in order to inform their acquisition and protection activities as one part of the larger process of achieving software assurance.

6.1 CWSS description

CWSS is organized into three *metric groups*: Base Finding, Attack Surface and Environmental, as shown in Figure 1. Each group contains multiple metrics – also known as *factors* – that are used to compute a CWSS score for a weakness.



X.1525(15) F01

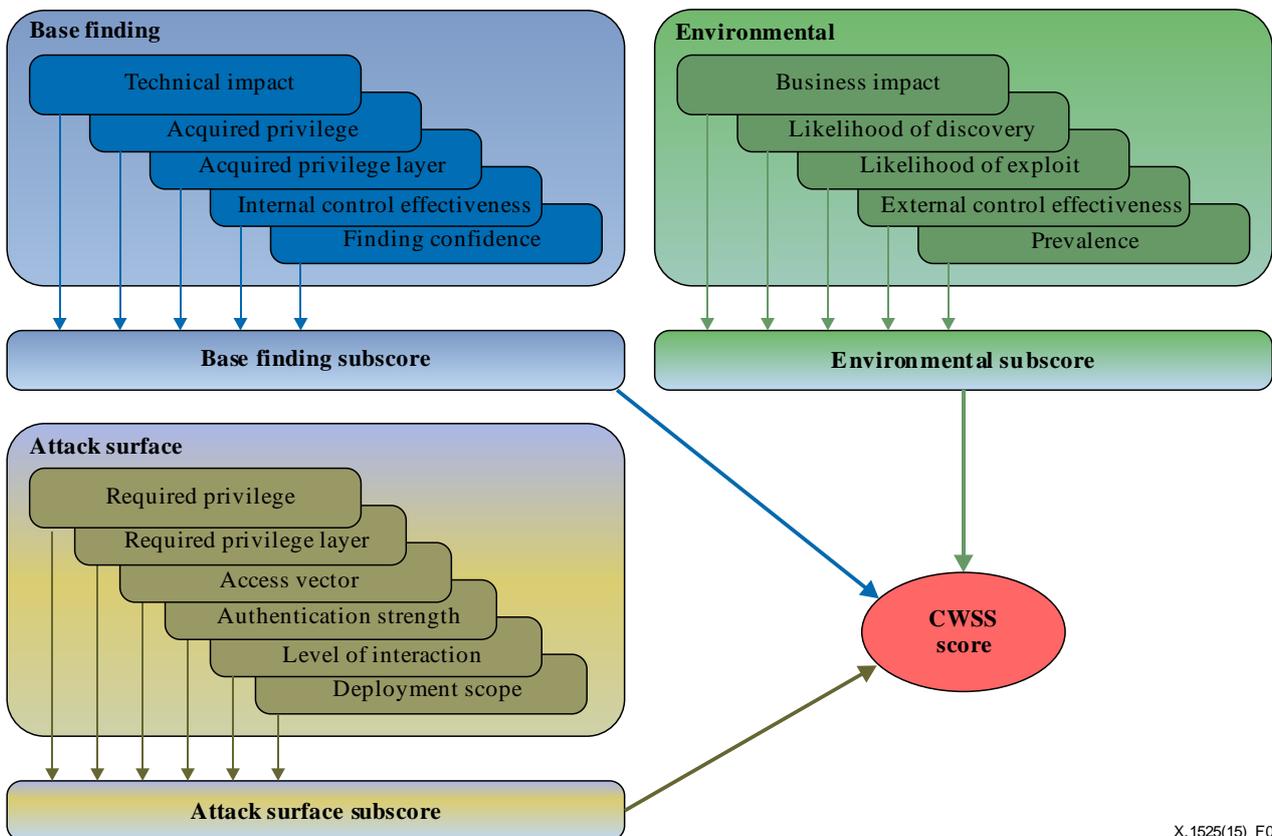
Figure 1 – CWSS metric groups

These metric groups are described as follows:

- Base Finding metric group: captures the inherent risk of the weakness, confidence in the accuracy of the finding and strength of controls. Base Finding metric group is discussed in clause 7.3.
- Attack Surface metric group: the barriers that an attacker must overcome in order to exploit the weakness. Attack Surface metric group is discussed in clause 7.4.
- Environmental metric group: characteristics of the weakness that are specific to a particular environment or operational context. Environmental metric group is discussed in clause 7.5.

6.2 Workings of CWSS

Each factor in the Base Finding metric group is assigned a value. These values are converted to associated weights, and a Base Finding subscore is calculated. The Base Finding subscore can range between 0 and 100. The same method is applied to the Attack Surface and Environmental metric group; their subscores can range between 0 and 1. Finally, the three subscores are multiplied together, which produces a CWSS score between 0 and 100, as illustrated below in Figure 2.



X.1525(15)_F02

Figure 2 – CWSS scoring

6.3 CWSS scoring

The stakeholder community is collaborating with MITRE to investigate several different scoring methods that might need to be supported within the CWSS framework. The current four scoring methods are:

- Targeted** Score individual weaknesses that are discovered in the design or implementation of a specific ("targeted") software package, e.g., a buffer overflow in the username of an authentication routine in line 1234 of server.c in an FTP server package. Automated tools and software security consultants use targeted methods when evaluating the security of a software package in terms of the weaknesses that are contained within the package.
- Generalized** Score classes of weaknesses independent of any particular software package, in order to prioritize them relative to each other (e.g., "buffer overflows are higher priority than memory leaks"). This approach is used by the CWE/SANS Top 25, OWASP Top Ten and similar efforts, but also by some automated code scanners. The generalized scores could vary significantly from the targeted scores that would result from a full analysis of the individual occurrences of the weakness class within a specific software package. For example, while the class of buffer overflows remains very important to many developers, individual buffer overflow bugs might be considered less important if they cannot be directly triggered by an attacker and their impact is reduced due to operating system (OS)-level protection mechanisms such as address space layout randomization (ASLR).
- Context-adjusted** Modify scores in accordance with the needs of a specific analytical context that may integrate business/mission priorities, threat environments, risk tolerance, etc. These needs are captured using vignettes that link inherent characteristics of weaknesses

with higher-level business considerations. This method could be applied to both targeted and generalized scoring.

Aggregated Combine the results of multiple, lower-level weakness scores to produce a single, overall score (or "grade"). While aggregation might be most applicable to the targeted method, it could also be used in generalized scoring, as occurred in the 2010 CWE/SANS Top 25.

Note that the current focus for most discussions about CWSS is on the Targeted scoring method and a framework for context-adjusted scoring. Methods for aggregated scoring will follow. Generalized scoring is being developed separately, primarily as part of the 2011 Top 25 and CWRAF.

CWSS scores can be automatically calculated, e.g., by a code analysis tool, or they can be manually calculated by a software security consultant or developer. Since automated analysis is not likely to have certain information available – such as the application's operating environment – CWSS scoring could possibly be conducted in multiple rounds: a tool first automatically calculates CWSS scores, then a human analyst manually adds additional details and recalculates the scores.

6.4 CWSS users

To be most effective, CWSS supports multiple usage scenarios by different stakeholders who all have an interest in a consistent scoring system for prioritizing software weaknesses that could introduce risks to products, systems, networks and services. Below are some examples of the primary stakeholders:

- **Software developers:** Developers often operate within limited time frames, due to release cycles and limited resources. As a result, they are unable to investigate and fix every reported weakness. They may choose to concentrate on the worst problems, the easiest-to-fix. In the case of automatic weakness findings, they might choose to focus on the findings that are least likely to be false positives.
- **Software development managers:** Development managers create strategies for prioritizing and removing entire classes of weaknesses from the entire code base, or at least the portion that is deemed to be most at risk, possibly by defining custom "Top-N" lists. They must understand the security implications of integrating third-party software, which may contain its own weaknesses. They may need to support distinct security requirements and prioritization for each product line.
- **Software acquirers:** Customers, including acquisition personnel, want to obtain third-party software with a reasonable level of assurance that the software provider has performed due diligence in removing or avoiding weaknesses that are most critical to the acquirer's business and mission. Related stakeholders include chief information officers (CIOs), chief security officers (CSOs), system administrators and end users of the software.
- **Enterprise security managers:** Enterprise security managers seek to minimize risk within their enterprise, both for well-known vulnerabilities in third-party products, as well as vulnerabilities (or weaknesses) in their own in-house software. They may wish to use a scoring mechanism that can be integrated with other security management processes, such as combining third-party vulnerability scanning results (for known third-party vulnerabilities) with custom application analysis (for in-house software) to help assess the overall risk to an asset.
- **Code analysis vendors and consultants:** Vendors and consultants often have their own custom scoring techniques, but they want to provide a consistent, community-vetted scoring mechanism for different customers.
- **Evaluators of code analysis capabilities:** Evaluators analyse and measure the capabilities of code analysis techniques (e.g., NIST SAMATE). They could use a consistent weakness

scoring mechanism to support sampling of reported findings, as well as understanding the severity of these findings without depending on ad hoc scoring methods that may vary widely by tool/technique.

- Other stakeholders: Other stakeholders may include vulnerability researchers, advocates of secure development and compliance-based analysts (e.g., PCI DSS).

As of June 2014 (when CWSS 0.8 was active), there are several real-world implementations of CWSS. The primary users have been code analysis vendors and software security consultants.

7 Metric groups

7.1 Metric group factors

CWSS contains the following factors, organized based on their metric group as shown in Table 1 below. Each factor is described in more detail in subsequent clauses.

Table 1 – Metric group factors

Group	Name	Summary
Base Finding	Technical impact (TI)	The potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited.
	Acquired privilege (AP)	The type of privileges that are obtained by an attacker who can successfully exploit the weakness.
	Acquired privilege layer (AL)	The operational layer to which the attacker gains privileges by successfully exploiting the weakness.
	Internal control effectiveness (IC)	The ability of the control to render the weakness unable to be exploited by an attacker.
	Finding confidence (FC)	The confidence that the reported issue is a weakness that can be utilized by an attacker
Attack Surface	Required privilege (RP)	The type of privileges that an attacker must already have in order to reach the code/functionality that contains the weakness.
	Required privilege layer (RL)	The operational layer to which the attacker must have privileges in order to attempt to attack the weakness.
	Access vector (AV)	The channel through which an attacker must communicate to reach the code or functionality that contains the weakness.
	Authentication strength (AS)	The strength of the authentication routine that protects the code/functionality that contains the weakness.
	Level of interaction (IN)	The actions that are required by the human victim(s) to enable a successful attack to take place.

Table 1 – Metric group factors

Group	Name	Summary
	Deployment scope (SC)	Whether the weakness is present in all deployable instances of the software, or if it is limited to a subset of platforms and/or configurations.
Environmental	Business impact (BI)	The potential impact to the business or mission if the weakness can be successfully exploited.
	Likelihood of discovery (DI)	The likelihood that an attacker can discover the weakness
	Likelihood of exploit (EX)	The likelihood that, if the weakness is discovered, an attacker with the required privileges/authentication/access would be able to successfully exploit it.
	External control effectiveness (EC)	The capability of controls or mitigations outside of the software that may render the weakness more difficult for an attacker to reach and/or trigger.
	Prevalence (P)	How frequently this type of weakness appears in software.

7.2 Values for uncertainty and flexibility

CWSS can be used in cases where there is little information at first, but the quality of information can improve over time. It is anticipated that in many use-cases, the CWSS score for an individual weakness finding may change frequently, as more information is discovered. Different entities may evaluate separate factors at different points in time.

As such, every CWSS factor effectively has "environmental" or "temporal" characteristics, so it is not particularly useful to adopt the same types of metric groups as are used in CVSS.

Most factors have in common the four values shown in Table 2.

Table 2 – Factor values for uncertainty and flexibility

Value	Usage
Unknown	<p>The entity calculating the score does not have enough information to provide a value for the factor. This can be a signal for further investigation. For example, an automated code scanner might be able to find certain weaknesses, but be unable to detect whether any authentication mechanisms are in place.</p> <p>The use of "Unknown" emphasizes that the score is incomplete or estimated, and further analysis may be necessary. This makes it easier to model incomplete information, and for the Business Value Context to influence final scores that were generated using incomplete information.</p> <p>The weight for this value is 0.5 for all factors, which generally produces a lower score; the addition of new information (i.e., changing some factors from "Unknown" to another value) will then adjust the score upward or downward based on the new information.</p>
Not applicable	The factor is being explicitly ignored in the score calculation. This effectively allows the Business Value Context to dictate whether a factor is relevant to the final score.

Table 2 – Factor values for uncertainty and flexibility

Value	Usage
	For example, a customer-focused CWSS scoring method might ignore the remediation effort, and a high-assurance environment might require investigation of all reported findings, even if there is low confidence in their accuracy. For a set of weakness findings for an individual software package, it is expected that all findings would have the same "Not applicable" value for the factor that is being ignored.
Quantified	The factor can be weighted using a quantified, continuous range of 0.0 through 1.0, instead of the factor's defined set of discrete values. Not all factors are quantifiable in this way, but it allows for additional customization of the metric.
Default	The factor's weight can be set to a default value. Labelling the factor as a default allows for investigation and possible modification at a later time.

7.3 Base finding metric group

The Base Finding metric group consists of the following factors:

- Technical impact (TI)
- Acquired privilege (AP)
- Acquired privilege layer (AL)
- Internal control effectiveness (IC)
- Finding confidence (FC)

The combination of values from technical impact, acquired privilege and acquired privilege layer gives the user some expressive power. For example, the user can characterize "High" technical impact with "Administrator" privilege at the "Application" layer.

7.3.1 Technical impact (TI)

Technical impact is the potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited. This is expressed in terms that are more fine-grained than confidentiality, integrity and availability.

The technical impact should be evaluated relative to the acquired privilege (AP) and acquired privilege layer (AL).

Table 3 – Technical impact weights

Value	Code	Weight	Description
Critical	C	1.0	Complete control over the software being analysed, to the point where operations cannot take place.
High	H	0.9	Significant control over the software being analysed, or access to critical information can be obtained.
Medium	M	0.6	Moderate control over the software being analysed, or access to moderately important information can be obtained.

Table 3 – Technical impact weights

Value	Code	Weight	Description
Low	L	0.3	Minimal control over the software being analysed, or only access to relatively unimportant information can be obtained.
None	N	0.0	There is no technical impact to the software being analysed at all. In other words, this does not lead to a vulnerability.
Default	D	0.6	The Default weight is the median of the weights for Critical, High, Medium, Low and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements; the user might want to investigate every weakness finding of interest, regardless of confidence.
Quantified	Q		This factor could be quantified with custom weights.

If this set of values is not precise enough, CWSS users can use their own Quantified methods to derive a subscore. One such method involves using the Common Weakness Risk Analysis Framework (CWRAF) [b-CWRAF] to define a vignette and a technical impact scorecard. The Impact weight is calculated using vignette-specific importance ratings for different technical impacts that could arise from exploitation of the weakness, such as modification of sensitive data, gaining privileges, resource consumption, etc.

7.3.2 Acquired privilege (AP)

The acquired privilege identifies the type of privileges that are obtained by an attacker who can successfully exploit the weakness.

Notice that the values are the same as those for required privilege, but the weights are different.

In some cases, the value for acquired privileges may be the same as for required privileges, which implies either (1) "horizontal" privilege escalation (e.g., from one unprivileged user to another) or (2) privilege escalation within a sandbox, such as an file transfer protocol (FTP)-only user who can escape to the shell.

Table 4 – Acquired privilege weights

Value	Code (Note)	Weight	Description
Administrator	A	1.0	The attacker gains access to an entity with administrator, root, SYSTEM, or equivalent privileges that imply full control over the software under analysis; or, the attacker can raise their own (lower) privileges to an administrator.
Partially-Privileged User	P	0.9	The attacker gains access to an entity with some special privileges, but not enough privileges that are equivalent to an administrator; or, the attacker can raise their own (lower) privileges to a partially-privileged user. For example, a user might have privileges to make backups, but not to modify the software's configuration or install updates.
Regular User	RU	0.7	The attacker gains access to an entity that is a regular user who has no special privileges; or, the attacker can raise their own (lower) privileges to that of a regular user.
Limited / Guest	L	0.6	The attacker gains access to an entity with limited or "guest" privileges that can significantly restrict allowable activities; or, the attacker can raise their own (lower) privileges to a guest. Note: this value does not refer to the "guest operating system" concept in virtualized hosts.
None	N	0.1	The attacker cannot gain access to any extra privileges beyond those that are already available to the attacker. (Note that this value can be useful in limited circumstances in which the attacker can escape a sandbox or other restrictive environment but still cannot gain extra privileges, or obtain access as other users.)
Default	D	0.7	Median of the weights for None, Guest, Regular User, Partially-privileged User and Administrator.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements that wants strict enforcement of privilege separation, even between already-privileged users.

Table 4 – Acquired privilege weights

Value	Code (Note)	Weight	Description
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since privileges and users are discrete entities, there might be limited circumstances in which a quantified model would be useful.
NOTE – A mnemonic for the main values in this factor is "RUNLAP" (<u>R</u> egular <u>U</u> ser, <u>N</u> one, <u>L</u> imited, <u>A</u> dm <u>i</u> n, <u>P</u> artially-Privileged).			

7.3.3 Acquired privilege layer (AL)

The acquired privilege layer identifies the operational layer to which the attacker gains privileges by successfully exploiting the weakness.

Table 5 – Acquired privilege layer weights

Value	Code (Note)	Weight	Description
Application	A	1.0	The attacker acquires privileges that are supported within the software under analysis itself. (If the software under analysis is an essential part of the underlying system, such as an operating system kernel, then the System value may be more appropriate.)
System	S	0.9	The attacker acquires privileges to the underlying system or physical host that is being used to run the software under analysis.
Network	N	0.7	The attacker acquires privileges to access the network.
Enterprise Infrastructure	E	1.0	The attacker acquires privileges to a critical piece of enterprise infrastructure, such as a router, switch, domain name system (DNS), domain controller, firewall, identity server, etc.
Default	D	0.9	Median of the weights for Application, System, Network and Enterprise Infrastructure.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not Applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements that wants strict enforcement of privilege separation, even between already-privileged users.
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since privilege layers are discrete entities, there might be limited circumstances in which a quantified model would be useful.
NOTE – A mnemonic for the main values in this factor is "SANE" (<u>S</u> ystem, <u>A</u> pplication, <u>N</u> etwork, <u>E</u> nterprise Infrastructure).			

7.3.4 Internal control effectiveness (IC)

An Internal Control is a control, protection mechanism, or mitigation that has been explicitly built into the software (whether through architecture, design, or implementation). Internal Control Effectiveness measures the ability of the control to render the weakness unable to be exploited by an attacker. For example, an input validation routine that restricts input length to 15 characters might be

moderately effective against cross site scripting (XSS) attacks by reducing the size of the XSS exploit that can be attempted.

When there are multiple internal controls, or multiple code paths that can reach the same weakness, then the following guidance applies:

- For each code path, analyse each internal control that exists along the code path, and choose the value with the lowest weight (i.e., the strongest internal control along the code path). This is called the code path value.
- Collect all code path values.
- Select the code path value that has the highest weight (i.e., is the weakest control).

This method evaluates each code path in terms of the code path's strongest control (since an attacker would have to bypass that control), then selects the weakest code path (i.e., the easiest route for the attacker to take).

Table 6 – Internal control effectiveness weights

Value	Code	Weight	Description
None	N	1.0	No controls exist
Limited	L	0.9	There are simplistic methods or accidental restrictions that might prevent a casual attacker from exploiting the issue
Moderate	M	0.7	The protection mechanism is commonly used but has known limitations that might be bypassed with some effort by a knowledgeable attacker. For example, the use of hypertext markup language (HTML) entity encoding to prevent XSS attacks may be bypassed when the output is placed into another context such as a cascading style sheet or HTML tag attribute.
Indirect (Defense-in-depth)	I	0.5	The control does not specifically protect against exploitation of the weakness, but it indirectly reduces the impact when a successful attack is launched, or otherwise makes it more difficult to construct a functional exploit. For example, a validation routine might indirectly limit the size of an input, which might make it difficult for an attacker to construct a payload for an XSS or structured query language (SQL) injection attack.
Best-available	B	0.3	The control follows best current practices, although it may have some limitations that can be overcome by a skilled, determined attacker, possibly requiring the presence of other weaknesses. For example, the double-submit method for cross-site-request-forgery (CSRF) protection is considered one of the strongest available, but it can be defeated in conjunction with behaviours of certain functionality that can read raw HTTP headers.
Complete	C	0.0	The control is completely effective against the weakness, i.e., there is no bug or vulnerability, and no adverse consequence of exploiting the issue. For example, a buffer copy operation that ensures that the destination buffer is always larger than the source (plus any indirect expansion of the original source size) will not cause an overflow.
Default	D	0.6	Median of the weights for Complete, Best-available, Indirect, Moderate, Limited and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.

Table 6 – Internal control effectiveness weights

Value	Code	Weight	Description
Not Applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses
Quantified	Q		This factor could be quantified with custom weights

7.3.5 Finding confidence (FC)

Finding confidence is the confidence that the reported issue:

1. is a weakness, and
2. can be triggered or utilized by an attacker.

Table 7 – Finding confidence weights

Value	Code	Weight	Description
Proven true	T	1.0	The weakness is reachable by the attacker
Proven locally true	LT	0.8	The weakness occurs within an individual function or component whose design relies on safe invocation of that function, but attacker reachability to that function is unknown or not present. For example, a utility function might construct a database query without encoding its inputs, but if it is only called with constant strings, the finding is locally true.
Proven false	F	0.0	The finding is erroneous (i.e., the finding is a false positive and there is no weakness) and/or there is no possible attacker role
Default	D	0.8	Median of the weights for Proven true, Proven locally true and Proven false.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements; the user might want to investigate every weakness finding of interest, regardless of confidence.
Quantified	Q		This factor could be quantified with custom weights. Some code analysis tools have precise measurements of the accuracy of specific detection patterns.

7.4 Attack Surface metric group

The Attack Surface metric group consists of the following factors:

- Required privilege (RP)
- Required privilege layer (RL)
- Access vector (AV)
- Authentication strength (AS)
- Level of interaction (IN)
- Deployment scope (SC).

7.4.1 Required privilege (RP)

The Required Privilege identifies the type of privileges that an attacker must already have in order to reach the code/functionality that contains the weakness.

Table 8 – Required privilege weights

Value	Code (Note)	Weight	Description
None	N	1.0	No privileges are required. For example, a web-based search engine may not require any privileges for an entity to enter a search term and view results.
Limited / Guest	L	0.9	The entity has limited or "guest" privileges that can significantly restrict allowed activities; the entity might be able to register or create a new account without any special requirements or proof of identity. For example, a web blog might allow participants to create a user name and submit a valid email address before entering comments. Note: this value does not refer to the "guest operating system" concept in virtualized hosts.
Regular user	RU	0.7	The entity is a regular user who has no special privileges.
Partially-privileged user	P	0.6	The entity is a valid user with some special privileges, but not enough privileges that are equivalent to an administrator. For example, a user might have privileges to make backups, but not to modify the software's configuration or install updates.
Administrator	A	0.1	The entity has administrator, root, SYSTEM, or equivalent privileges that imply full control over the software or the underlying OS.
Default	D	0.7	Median of the weights for None, Limited, Regular user, Partially-privileged user and Administrator.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements that wants strict enforcement of privilege separation, even between already-privileged users.

Table 8 – Required privilege weights

Value	Code (Note)	Weight	Description
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since privileges and users are discrete entities, there might be limited circumstances in which a quantified model would be useful.
NOTE – A mnemonic for the main values in this factor is "RUNLAP" (<u>R</u> egular <u>U</u> ser, <u>N</u> one, <u>L</u> imited, <u>A</u> admin, <u>P</u> artially-Privileged).			

7.4.2 Required privilege layer (RL)

The required privilege Layer identifies the operational layer to which the attacker must have privileges in order to attempt to attack the weakness.

Table 9 – Required privilege layer weights

Value	Code (Note)	Weight	Description
Application	A	1.0	The attacker must have privileges that are supported within the software under analysis itself. (If the software under analysis is an essential part of the underlying system, such as an operating system kernel, then the System value may be more appropriate.)
System	S	0.9	The attacker must have privileges to the underlying system or physical host that is being used to run the software under analysis.
Network	N	0.7	The attacker must have privileges to access the network.
Enterprise infrastructure	E	1.0	The attacker must have privileges on a critical piece of enterprise infrastructure, such as a router, switch, DNS, domain controller, firewall, identity server, etc.
Default	D	0.9	Median of the weights for Application, System, Network and Enterprise Infrastructure.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements that wants strict enforcement of privilege separation, even between already-privileged users.
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since privilege layers are discrete entities, there might be limited circumstances in which a quantified model would be useful.
NOTE – A mnemonic for the main values in this factor is "SANE" (<u>S</u> ystem, <u>A</u> pplication, <u>N</u> etwork, <u>E</u> nterprise Infrastructure).			

7.4.3 Access vector (AV)

The access vector identifies the channel through which an attacker must communicate to reach the code or functionality that contains the weakness. Note that these values are very similar to the ones used in CVSS, except CWSS distinguishes between physical access and local (shell/account) access.

While there is a close relationship between access vector and required privilege layer, the two are distinct. For example, an attacker with "physical" access to a router might be able to affect the network or enterprise layer.

Table 10 – Access vector weights

Value	Code	Weight	Description
Internet	I	1.0	An attacker must have access to the Internet to reach the weakness.
Intranet	R	0.8	An attacker must have access to an enterprise intranet that is shielded from direct access from the Internet, e.g., by using a firewall, but otherwise the intranet is accessible to most members of the enterprise.
Private network	V	0.8	An attacker must have access to a private network that is only accessible to a narrowly-defined set of trusted parties.
Adjacent network	A	0.7	An attacker must have access to a physical interface to the network, such as the broadcast or collision domain of the vulnerable software. Examples of local networks include local Internet protocol (IP) subnet, Bluetooth, IEEE 802.11 and local Ethernet segment.
Local	L	0.5	The attacker must have an interactive, local (shell) account that interfaces directly with the underlying operating system.
Physical	P	0.2	The attacker must have physical access to the system that the software runs on, or otherwise able to interact with the system via interfaces such as universal serial bus (USB), compact disc (CD), keyboard, mouse, etc.
Default	D	0.75	Median of weights for relevant values.
Unknown	U	0.5	
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses.
Quantified	Q		This factor could be quantified with custom weights. Note that Quantified values are supported for completeness; however, since access vectors are discrete entities, there might be limited circumstances in which a quantified model would be useful.

7.4.4 Authentication strength (AS)

The Authentication strength covers the strength of the authentication routine that protects the code/functionality that contains the weakness.

When more than one authentication routine is in use, or if two or more code paths exist, the scoring should be performed as follows:

When there are multiple authentication routines, or multiple code paths that can reach the same weakness, then the following guidance applies:

- For each code path, analyse each authentication routine that exists along the code path, and choose the value with the lowest weight (i.e., the strongest authentication routine along the code path). This is called the code path value.
- Collect all code path values.
- Select the code path value that has the highest weight (i.e., contains the weakest routine).

This method evaluates each code path in terms of the code path's strongest authentication routine (since an attacker would have to bypass that control), then selects the weakest code path (i.e., the easiest route for the attacker to take).

Table 11 – Authentication strength weights

Value	Code	Weight	Description
Strong	S	0.7	The weakness requires strongest-available methods to tie the entity to a real-world identity, such as hardware-based tokens and/or multi-factor authentication.
Moderate	M	0.8	The weakness requires authentication using moderately strong methods, such as the use of certificates from untrusted authorities, knowledge-based authentication, or one-time passwords.
Weak	W	0.9	The weakness requires a simple, weak authentication method that is easily compromised using spoofing, dictionary, or replay attacks, such as a static password.
None	N	1.0	The weakness does not require any authentication at all.
Default	D	0.85	Median of values for Strong, Moderate, Weak and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This might not be applicable in an environment with high assurance requirements that seek to eliminate all weaknesses.
Quantified	Q		This factor could be quantified with custom weights.

7.4.5 Level of interaction (IN)

The Level of Interaction covers the actions that are required by the human victim(s) to enable a successful attack to take place.

Table 12 – Level of interaction weights

Value	Code	Weight	Description
Automated	A	1.0	No human interaction is required.
Typical/Limited	T	0.9	The attacker must convince the user to perform an action that is common or regarded as "normal" within typical product operation. For example, clicking on a link in a web page, or previewing the body of an email, is common behaviour.
Moderate	M	0.8	The attacker must convince the user to perform an action that might appear suspicious to a cautious, knowledgeable user. For example: the user has to accept a warning that suggests the attacker's payload might contain dangerous content.
Opportunistic	O	0.3	The attacker cannot directly control or influence the victim, and can only passively capitalize on mistakes or actions of others.
High	H	0.1	A large amount of social engineering is required, possibly including ignorance or negligence on the part of the victim.

Table 12 – Level of interaction weights

Value	Code	Weight	Description
No interaction	NI	0.0	There is no interaction possible, not even opportunistically; this typically would render the weakness as a "bug" instead of leading to a vulnerability. Since CWSS is for security, the weight is 0.
Default	D	0.55	Median of values for Automated, Limited, Moderate, Opportunistic, High and No interaction.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.

7.4.6 Deployment scope (SC)

Deployment scope identifies whether the weakness is present in all deployable instances of the software, or if it is limited to a subset of platforms and/or configurations. For example, a numeric calculation error might only be applicable for software that is running under a particular OS and a 64-bit architecture, or a path traversal issue might only affect operating systems for which "\" is treated as a directory separator.

Table 13 – Deployment scope weights

Value	Code (Note 1)	Weight	Description
All	A	1.0	Present in all platforms or configurations
Moderate	M	0.9	Present in common platforms or configurations
Rare	R	0.5	Only present in rare platforms or configurations
Potentially reachable	P	0.1	Potentially reachable (Note 2), but all code paths are currently safe, and/or the weakness is in dead code
Default	D	0.7	The median of weights for RAMP values
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses
Quantified	Q		This factor could be quantified with custom weights. The user may know what percentage of shipped (or supported) software contains this bug.

NOTE 1 – A mnemonic for the main values in this factor is "RAMP" (Rare, All, Moderate, Potentially Reachable).

NOTE 2 – "Potentially Reachable" has some overlap with "Locally True" in the Finding Confidence (FC) factor.

7.5 Environmental Metric group

The Environmental Metric group consists of the following factors:

- Business impact (BI)
- Likelihood of discovery (DI)
- Likelihood of exploit (EX)

- External control effectiveness (EC)
- Prevalence (P).

7.5.1 Business impact (BI)

Business impact describes the potential impact to the business or mission if the weakness can be successfully exploited.

NOTE – Since business concerns vary widely across organizations, CWSS 1.0 does not attempt to provide a more precise breakdown, e.g., in terms of financial, reputational, physical, legal or other types of damage. This factor can be quantified to support any externally-defined models.

Table 14 – Business impact weights

Value	Code	Weight	Description
Critical	C	1.0	The business/mission could fail
High	H	0.9	The operations of the business/mission would be significantly affected
Medium	M	0.6	The business/mission would be affected, but without extensive damage to regular operations.
Low	L	0.3	Minimal impact on the business/mission
None	N	0.0	No impact
Default	D	0.6	The median of weights for Critical, High, Medium, Low and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in contexts in which the business impact is irrelevant, or if the impact is being assessed and considered in analytical processes that are outside of the CWSS score itself.
Quantified	Q		This factor could be quantified with custom weights. Some organizations might have specific measurements for the business value of the asset, for example, which could be integrated into this measurement.

7.5.2 Likelihood of discovery (DI)

Likelihood of discovery is the likelihood that an attacker can discover the weakness.

NOTE – This factor was considered for removal in CWSS 1.0, since it can be difficult to measure and can be influenced by other factors such as acquired privilege, technical impact and prevalence. However, it has been preserved to reflect that some developers will use likelihood of discovery to help prioritize how quickly an issue should be fixed.

Table 15 – Likelihood of discovery weights

Value	Code	Weight	Description
High	H	1.0	It is very likely that an attacker can discover the weakness quickly and with little effort using simple techniques, without access to source code or other artefacts that simplify weakness detection.
Medium	M	0.6	An attacker might be able to discover the weakness, but would require certain skills to do so, possibly requiring source code access or reverse engineering knowledge. It may require some time investment to find the issue.
Low	L	0.2	An attacker is unlikely to discover the weakness without highly specialized skills, access to source code (or its equivalent) and a large time investment.
Default	D	0.6	The median of the High, Medium and Low values.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This might not be applicable when the scorer assumes that all weaknesses will be discovered by an attacker.
Quantified	Q		This factor could be quantified with custom weights

7.5.3 Likelihood of exploit (EX)

Likelihood of exploit is the likelihood that, if the weakness is discovered, an attacker with the required privileges/authentication/access would be able to successfully exploit it.

Table 16 – Likelihood of exploit weights

Value	Code	Weight	Description
High	H	1.0	It is highly likely that an attacker would target this weakness successfully, with a reliable exploit that is easy to develop.
Medium	M	0.6	An attacker would probably target this weakness successfully, but the chances of success might vary, or require multiple attempts to succeed.
Low	L	0.2	An attacker probably would not target this weakness, or could have very limited chances of success.
None	N	0.0	An attacker has no chance of success; i.e., the issue is a "bug" because there is no attacker role, and no benefit to the attacker.
Default	D	0.6	Median of the High, Medium and Low values. The "None" value is ignored with the expectation that few weakness findings would be scored using the value, and including it in the median calculation would reduce the weight to a non-intuitive level.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.

Table 16 – Likelihood of exploit weights

Value	Code	Weight	Description
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. For example, the scorer might want to assume that attackers could exploit any weakness they can find, or be willing to invest significant resources to work around any possible barriers to exploit success.
Quantified	Q		This factor could be quantified with custom weights

Note that this factor is influenced by the Impact of a weakness, since attackers often target weaknesses that have the most severe impacts. Alternately, they may target weaknesses that are easy to trigger. It is also influenced by other factors such as the effectiveness of internal and external controls.

It might seem that the prevalence is also an influence, but prevalence is more closely related to likelihood of discovery.

7.5.4 External control effectiveness (EC)

External control effectiveness is the capability of controls or mitigations outside of the software that may render the weakness more difficult for an attacker to reach and/or trigger. For example, address space layout randomization (ASLR) and similar technologies reduce, but do not eliminate, the chances of success for a buffer overflow attack. However, ASLR is not directly instantiated within the software itself.

When there are multiple external controls, or multiple code paths that can reach the same weakness, then the following guidance applies:

- For each code path, analyse each external control that exists along the code path and choose the value with the lowest weight (i.e., the strongest external control along the code path). This is called the code path value.
- Collect all code path values.
- Select the code path value that has the highest weight (i.e., is the weakest control).

This method evaluates each code path in terms of the code path's strongest control (since an attacker would have to bypass that control), then selects the weakest code path (i.e., the easiest route for the attacker to take).

Table 17 – External control effectiveness weights

Value	Code	Weight	Description
None	N	1.0	No controls exist
Limited	L	0.9	There are simplistic methods or accidental restrictions that might prevent a casual attacker from exploiting the issue
Moderate	M	0.7	The protection mechanism is commonly used but has known limitations that might be bypassed with some effort by a knowledgeable attacker
Indirect (Defense-in-depth)	I	0.5	The control does not specifically protect against exploitation of the weakness, but it indirectly reduces the impact when a successful attack is launched, or otherwise makes it more difficult to construct a functional exploit. For example, address space layout randomization (ASLR) and similar technologies reduce, but do not eliminate, the chances of success in a buffer overflow attack. Since the response is typically to exit the process, the result is still a denial of service.
Best-available	B	0.3	The control follows best current practices, although it may have some limitations that can be overcome by a skilled, determined attacker, possibly requiring the presence of other weaknesses. For example, transport layer security (TLS) / secure sockets layer (SSL 3) are in operation throughout much of the web, and stronger methods generally are not available due to compatibility issues.
Complete	C	0.1	The control is completely effective against the weakness, i.e., there is no bug or vulnerability, and no adverse consequence of exploiting the issue. For example, a sandbox environment might restrict file access operations to a single working directory, which would protect against exploitation of path traversal. A non-zero weight is used to reflect the possibility that the external control could be accidentally removed in the future, e.g., if the software's environment changes.
Default	D	0.6	The median of Complete, Best-Available, Indirect, Moderate, Limited and None.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses
Quantified	Q		This factor could be quantified with custom weights

7.5.5 Prevalence (P)

The prevalence of a finding identifies how frequently this type of weakness appears in software.

NOTE – This factor might be considered for removal in future versions. However, it is too closely tied to generalized scoring methods and CWRAF to be removed within CWSS 1.0.

This factor is intended for use in generalized scoring of classes of weaknesses, such as the development of custom Top-N weakness lists. When scoring an individual weakness finding in an automated-scanning context, this factor is likely to use a "Not applicable" value.

Table 18 – Prevalence weights

Value	Code	Weight (Note)	Description
Widespread	W	1.0	The weakness is found in most or all software in the associated environment, and may occur multiple times within the same software package.
High	H	0.9	The weakness is encountered very often, but it is not widespread.
Common	C	0.8	The weakness is encountered periodically
Limited	L	0.7	The weakness is encountered rarely, or never.
Default	D	0.85	The median of Limited, Common, High and Widespread.
Unknown	UK	0.5	There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.
Not applicable	NA	1.0	This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. When performing targeted scoring against specific weakness findings in an application, Prevalence is normally expected to be irrelevant, since the individual application and the analytical techniques determine how frequently the weakness occurs, and many aggregated scoring methods will generate larger scores if there are more weaknesses.
Quantified	Q		This factor could be quantified with custom weights. Precise prevalence data may be available within limited use cases, provided the user is tracking weakness data at a low level of granularity. For example, a developer may be tracking weaknesses across a suite of products, or a code-auditing vendor could measure prevalence from the software analysed across the entire customer base. In a previous version of CWSS, prevalence was calculated based on raw voting data that was collected for the 2010 Top 25, which used discrete values (range 1 to 4) which were then adjusted to a 1-to-10 range.
NOTE – Since software can be successfully attacked even in the presence of a single weakness, the selected weights do not provide significant distinction between each other.			

7.6 CWSS score formula

A CWSS 1.0 score can range between 0 and 100. It is calculated as follows:

$$\text{BaseFindingSubscore} * \text{AttackSurfaceSubscore} * \text{EnvironmentSubscore}$$

The BaseFindingSubscore supports values between 0 and 100. Both the AttackSurfaceSubscore and EnvironmentSubscore support values between 0 and 1.

7.6.1 Base finding subscore

The Base finding subscore (BaseFindingSubscore) is calculated as follows:

$$\text{Base} = [(10 * \text{TechnicalImpact} + 5 * (\text{AcquiredPrivilege} + \text{AcquiredPrivilegeLayer}) + 5 * \text{FindingConfidence}) * f(\text{TechnicalImpact}) * \text{InternalControlEffectiveness}] * 4.0$$

$$f(\text{TechnicalImpact}) = 0 \text{ if } \text{TechnicalImpact} = 0; \text{ otherwise } f(\text{TechnicalImpact}) = 1.$$

The maximum potential BaseFindingSubscore is 100.

The definition of f(TechImpact) has an equivalent in CVSS. It is used to ensure that if the Technical Impact is 0, that the other added factors do inadvertently generate a non-zero score.

TechnicalImpact and the AcquiredPrivilege/AcquiredPrivilegeLayer combination are given equal weight, each accounting for 40% of the BaseFindingSubscore. (Each generate a sub-value with a maximum of 10). There is some adjustment for Finding Confidence, which accounts for 20% of the Base (maximum of 5). The InternalControlEffectiveness can adjust the score downward, perhaps to 0, depending on the strength of any internal controls that have been applied to the issue. After application of InternalControlEffectiveness, the possible range of results is between 0 and 25, so the 4.0 coefficient is used to adjust the BaseFindingSubscore to a range between 0 and 100.

7.6.2 Attack surface subscore

The AttackSurfaceSubscore is calculated as:

$$[20*(RequiredPrivilege + RequiredPrivilegeLayer + AccessVector) + 20*DeploymentScope + 15*LevelOfInteraction + 5*AuthenticationStrength] / 100.0$$

The combination of required privileges / access makes up 60% of the Attack Surface subscore; deployment scope, another 20%; interaction, 15%; and authentication, 5%. The authentication requirements are not given much focus, under the assumption that strong proof of identity will not significantly deter an attacker from attempting to exploit the vulnerability.

This generates a range of values between 0 and 100, which are then divided by 100.

7.6.3 Environmental subscore

The EnvironmentalSubscore is calculated as:

$$[(10*BusinessImpact + 3*LikelihoodOfDiscovery + 4*LikelihoodOfExploit + 3*Prevalence) * f(BusinessImpact) * ExternalControlEffectiveness] / 20.0$$

$$f(BusinessImpact) = 0 \text{ if } BusinessImpact == 0; \text{ otherwise } f(BusinessImpact) = 1$$

BusinessImpact accounts for 50% of the environmental score, and it can move the final score to 0. ExternalControlEffectiveness is always non-zero (to account for the risk that it can be inadvertently removed if the environment changes), but otherwise it can have major impact on the final score. The combination of LikelihoodOfDiscovery and LikelihoodOfExploit accounts for 35% of the score, and Prevalence at 15%.

7.6.4 Additional features of the formula

There is significant diversity in the kinds of scores that can be represented, although the use of multiplication of many different factors, combined with multiple weights with small values, means that the range of potential scores is somewhat skewed towards lower values.

Since "Not Applicable" values have a weight of 1, the formula always has a potential maximum score of 100.0. In extremely rare cases in which certain factors are treated as Not applicable (e.g., technical impact, business impact and internal control effectiveness), then the minimum possible score could be non-zero.

When default values are used for a large number of factors for a single score, using the median weights as defined in CWSS 1.0, the scores will skew heavily to the low side. The median weight for a factor does not necessarily reflect the most likely value that could be used, so the selection of default weights may be changed in future versions. Ideally, the formula would have a property in which the use of many default values produces a score that is relatively close to 50; the selection of non-default values would adjust the final score upward or downward, thereby increasing precision.

The use of "Unknown" values also generally produces scores that skew to the low side. This might be a useful feature, since scores will be higher if they have more specific information.

7.7 CWSS vectors, scoring examples and score portability

Using the Codes as specified for each factor, a CWSS score can be stored in a compact, machine-parsable, human-readable format that provides the details for how the score was generated. This is very similar to how CVSS vectors are constructed.

Unlike CVSS, not all CWSS factors can be described symbolically with discrete values. Any factor can be quantified with continuous weights that override the originally-defined default discrete values, using the "Q" value. When calculated using CWRAF, the Impact factor is effectively an expression of 32 separate Technical Impacts and layers, many of which would not be applicable to a particular weakness. Treating each impact as a separate factor would roughly double the number of factors required to calculate a CWSS score. In addition, CWRAF's use of business value context (BVC) to adjust scores for business-specific concerns also means that a CWSS score and its vector may appear to be inconsistent if they are "transported" to other domains or vignettes.

With this concern in mind, a CWSS 1.0 vector should explicitly list the weights for each factor, even though it increases the size of the vector representation.

The format of a single factor in a CWSS vector is:

FactorName:Value,Weight

For example, "P:NA,1.0" specifies a "Not Applicable" value for Prevalence with a weight of 1.0. A specifier of "AV:P,0.2" indicates the "Physical" value for Access Vector with a weight of 0.2.

Factors are separated by forward slash characters, such as:

```
AV:I,1.0/RP:G,0.9/AS:N,1.0
```

which lists values and weights for "AV" (Access Vector), "RP" (Required Privilege Level) and "AS" (Authentication Strength).

If a CWSS vector is provided that does not list the actual weights for a value, then an implementation should report a possible error or inconsistency, try to infer the CWSS version based on the vector's factors and values, re-calculate the CWSS score based on the inferred version, and compare this to the original score. If the scores are inconsistent, the implementation should report a possible error or inconsistency.

7.7.1 Example: Business-critical application

Consider a reported weakness in which an application is the primary source of income for a company, thus has critical business value. The application allows arbitrary Internet users to sign up for an account using only an email address. A user can then exploit the weakness to obtain administrator privileges for the application, but the attack cannot succeed until the administrator views a report of recent user activities – a common occurrence. The attacker cannot take complete control over the application, but can delete its users and data. Suppose further that there are no controls to prevent the weakness, but the fix for the issue is simple, and limited to a few lines of code.

This situation could be captured in the following CWSS vector:

```
(TI:H,0.9/AP:A,1.0/AL:A,1.0/IC:N,1.0/FC:T,1.0/  
RP:G,0.9/RL:A,1.0/AV:I,1.0/AS:N,1.0/IN:T,0.9/SC:A,1.0/  
BI:C/0.9,DI:H,1.0/EX:H,1.0/EC:N,1.0/P:NA,1.0)
```

The vector has been split into multiple lines for readability. Each line represents a metric group.

The factors and values are as shown in Table 19, below.

Table 19 – Factors and values for business-critical application example

Factor	Value
Technical Impact	High
Acquired privilege	Administrator
Acquired privilege Layer	Application
Internal control effectiveness	None
Finding confidence	Proven true
Required privilege	Guest
Required Privilege layer	Application
Access vector	Internet
Authentication strength	None
Level of interaction	Typical/Limited
Deployment scope	All
Business impact	Critical
Likelihood of discovery	High
Likelihood of exploit	High
External control effectiveness	None
Prevalence	NA

The CWSS score for this vector is 92.6, derived as follows:

- **BaseSubscore:**
 - $[(10 * TI + 5*(AP + AL) + 5*FC) * f(TI) * IC] * 4.0$
 - $f(TI) = 1$
 - $= [(10 * 0.9 + 5*(1.0 + 1.0) + 5*1.0) * 1 * 1.0] * 4.0$
 - $= [(9.0 + 10.0 + 5.0) * 1.0] * 4.0$
 - $= 24.0 * 4.0$
 - $= 96.0$
- **AttackSurfaceSubscore:**
 - $[20*(RP + RL + AV) + 20*SC + 15*IN + 5*AS] / 100.0$
 - $= [20*(0.9 + 1.0 + 1.0) + 20*1.0 + 15*0.9 + 5*1.0] / 100.0$
 - $= [58.0 + 20.0 + 13.5 + 5.0] / 100.0$
 - $= 96.5 / 100.0$
 - $= 0.965$
- **EnvironmentSubscore:**
 - $[(10*BI + 3*DI + 4*EX + 3*P) * f(BI) * EC] / 20.0$
 - $f(BI) = 1$
 - $= [(10*1.0 + 3*1.0 + 4*1.0 + 3*1.0) * 1 * 1.0] / 20.0$
 - $= [(10.0 + 3.0 + 4.0 + 3.0) * 1.0] / 20.0$
 - $= 20.0 / 20.0$
 - $= 1.0$

The final score is:

$$96.0 * 0.965 * 1.0 = 92.64 == 92.6$$

7.7.2 Example: Wiki with limited business criticality

Consider this CWSS vector. Suppose the software is a wiki that is used for tracking social events for a mid-size business. Some of the most important characteristics are that there is medium technical impact to an application administrator from a regular user of the application, but the application is not business-critical, so the overall business impact is low. Also note that most of the environment factors are set to "Not applicable."

(TI:M,0.6/AP:A,1.0/AL:A,1.0/IC:N,1.0/FC:T,1.0/
 RP:RU,0.7/RL:A,1.0/AV:I,1.0/AS:W,0.9/IN:A,1.0/SC:NA,1.0/
 BI:L/0.3,DI:NA,1.0/EX:NA,1.0/EC:N,1.0/RE:NA,1.0/P:NA,1.0)

The vector has been split into multiple lines for readability. Each line represents a metric group. The factors and values are as shown in Table 20, below.

Table 20 – Factors and values for limited business-critical application example

Factor	Value
Technical impact	Medium
Acquired privilege	Administrator
Acquired privilege layer	Application
Internal control effectiveness	None
Finding confidence	Proven true
Required privilege	Regular user
Required privilege layer	Application
Access vector	Internet
Authentication strength	Weak
Level of interaction	Automated
Deployment scope	NA
Business impact	Low
Likelihood of discovery	NA
Likelihood of exploit	NA
External control effectiveness	None
Prevalence	NA

The CWSS score for this vector is 51.1, derived as follows:

- **BaseSubscore:**
 - $[(10 * TI + 5*(AP + AL) + 5*FC) * f(TI) * IC] * 4.0$
 - $f(TI) = 1$
 - $= [(10 * 0.6 + 5*(1 + 1) + 5*1) * f(TI) * 1] * 4.0$
 - $= 84.0$
- **AttackSurfaceSubscore:**
 - $[20*(RP + RL + AV) + 20*SC + 15*IN + 5*AS] / 100.0$
 - $= [20*(0.7 + 1 + 1) + 20*1.0 + 15*1.0 + 5*0.9] / 100.0$
 - $= [54.0 + 20.0 + 15.0 + 4.5] / 100.0$
 - $= 93.5 / 100.0$
 - $= 0.94 (0.935)$

- **EnvironmentSubscore:**
 - $[(10*BI + 3*DI + 4*EX + 3*P) * f(BI) * EC] / 20.0$
 - $f(BI) = 1$
 - $= [(10*0.3 + 3*1.0 + 4*1.0 + 3*1.0) * f(BI) * 1] / 20.0$
 - $= [(3.0 + 3.0 + 4.0 + 3.0) * 1.0 * 1.0] / 20.0$
 - $= [13.0 * 1.0] / 20.0$
 - $= 0.65$

The final score is:

$$84.0 * 0.935 * 0.65 = 51.051 == 51.1$$

7.7.3 Other approaches to CWSS score portability

Instead of recording each individual weight within a CWSS vector, several other methods could be adopted.

One possibility is to extend the CWSS vectors to record additional metadata that does not affect the score but reflects the version or other important information. The metadata portion would not necessarily need to capture weights, per se. For example, the CWSS version could be recorded by using a "factor" name such as "V" along with a value that represents the CWSS version, e.g., "V:1.1". This would add approximately 4 bytes to each CWSS vector. However, if the version is encoded within a vector, then the assigned weights would no longer need to be recorded (except for Quantified values), so the resulting vectors could be much shorter.

A different approach would be to attach metadata to a set of generated CWSS scores (such as the Technical Impact Scorecard if CWRAF is used), but it could be too easy for this metadata to become detached from the scores/vectors. Quantified factors would still need to be represented within a vector, since they could vary for each weakness finding.

Another approach is that when CWSS scores are transferred from one party to the other, then the receiving party could re-calculate the scores from the given CWSS vectors, then compare the re-calculated scores with the original scores. A difference in scores would suggest that different mechanisms are in use between the provider and receiver, possibly a different CWSS version.

Bibliography

- [b-ITU-T X.1500] Recommendation ITU-T X.1500 (2011), *Overview of Cybersecurity information exchange*.
- [b-ITU-T X.1520] Recommendation ITU-T X.1520 (2014), *Common vulnerabilities exposures*.
- [b-ITU-T X.1521] Recommendation ITU-T X.1521 (2011), *Common vulnerability scoring system*.
- [b-ITU-T X.1524] Recommendation ITU-T X.1524 (2012), *Common weakness enumeration*.
- [b-ITU-T X.1544] Recommendation ITU-T X.1544 (2013), *Common attack pattern enumeration and classification*.
- [b-CWRAF] Common Weakness Risk Analysis Framework
<http://cwe.mitre.org/cwraf/>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems