Common Vulnerability Scoring System v3.0: User Guide

The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities. CVSS consists of three metric groups: Base, Temporal, and Environmental. The Base group represents the intrinsic qualities of a vulnerability, the Temporal group reflects the characteristics of a vulnerability that change over time, and the Environmental group represents the characteristics of a vulnerability that are unique to a user's environment. The Base metrics produce a score ranging from 0.0 to 10.0, which can then be modified by scoring the Temporal and Environmental metrics. A CVSS score is also represented as a vector string, a compressed textual representation of the values used to derive the score. This document provides a guide to scoring vulnerabilities using the CVSS v3.0 standard.

Contents

Resources & Links

Below, are useful references to additional CVSS v3.0 documents.

Resource	Location
Specification Document	Includes metric descriptions, formulas, and vector string. Available at, https://www.first.org/cvss/specification-document
User guide	Includes further discussion of CVSS v3.0, a scoring rubric, and a glossary. Available at, https://www.first.org/cvss/user-guide
Example document	Includes helpful examples of CVSS v3.0 scoring in. practice. Available at, https://www.first.org/cvss/examples
CVSS v3.0 logo	Low and hi-res images available at, https://www.first.org/cvss/identity
CVSS v3.0 calculator	Reference implementation of the CVSS v3.0 equations, available at, https://www.first.org/cvss/calculator/3.0
JSON and XML schemas	JSON and XML schema definitions available at, https://www.first.org/cvss/data-representations

Introduction

This guide supplements the formal CVSS v3.0 specification document by providing additional information, highlighting relevant changes from v2.0, as well as providing scoring guidance and a scoring rubric.

The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as a textual representation of that score. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes.

CVSS affords three important benefits:

- It provides standardized vulnerability scores. When an organization uses a common algorithm for scoring vulnerabilities across all IT platforms, it can leverage a single vulnerability management policy defining the maximum allowable time to validate and remediate a given vulnerability.

- It provides an open framework. Users may be confused when a vulnerability is assigned an arbitrary score by a third party. With CVSS, the individual characteristics used to derive a score are transparent.

- CVSS helps prioritize risk. When the environmental score is computed, the vulnerability becomes contextual to each organization, and helps provide a better understanding of the risk posed by a vulnerability to the organization.

Since its initial release in 2004, CVSS has enjoyed widespread adoption. In September 2007, CVSS v2.0 was adopted as part of the Payment Card Industry Data Security Standard (PCI DSS). In order to comply with PCI DSS, merchants processing credit cards must demonstrate that none of their computing systems has a vulnerability with a CVSS score greater than or equal to 4.0. In 2007 NIST included CVSS v2.0 as part of their Security Content Automation Protocol (SCAP).[1] In April 2011, CVSS v2.0 was formally adopted as an international standard for scoring vulnerabilities (ITU-T X.1521).[2]

Changes in CVSS v3.0

Given the widespread adoption of CVSS v2.0, a number of opportunities for improvement had been identified, prompting the development of v3.0. These are described in detail below.

Scope, Vulnerable Component, and Impacted Component

CVSS v2.0 presented difficulties for vendors when scoring vulnerabilities that would fully compromise their software, but only partially affect the host operating system. In v2.0 vulnerabilities are scored relative to the host operating system, which led one application vendor to adopt a “Partial+” impact metric convention.[3] CVSS v3.0 addresses this issue with updates to

1 See http://scap.nist.gov/.

2 See https://www.itu.int/rec/T-REC-X.1521-201104-I/en.

3 For example, see http://www.oracle.com/technetwork/topics/security/cvssscoringsystem-091884.html.

where the impact metrics are scored and a new metric called Scope (discussed further below). Therefore, an important conceptual change in CVSS v3.0 is the ability to score vulnerabilities that exist in one software component (that we refer to formally as the *vulnerable component*) but which impact a separate software, hardware, or networking component (that we refer to formally as the *impacted component*), as illustrated in Figure 1.[4]

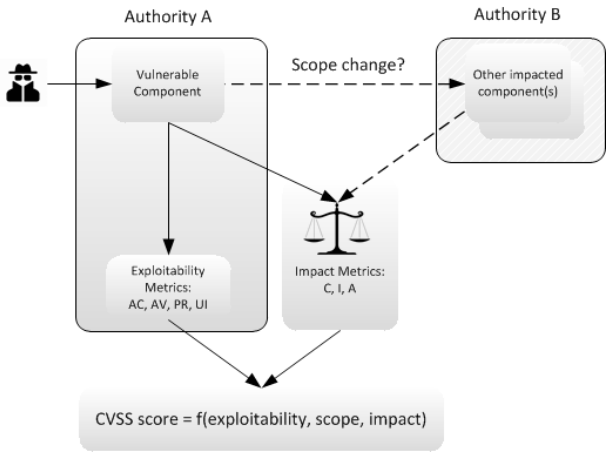


Figure 1: Scope change

For example, consider a vulnerability in a virtual machine that compromises the host operating system. The vulnerable component is the virtual machine, while the impacted component is the host operating system. Because these two components independently manage privileges to computing resources, they therefore represent separate (authorization) authorities. In Figure 1, the virtual machine is managed by "Authority A," while the host OS is managed by "Authority B." When two authorities are involved in a vulnerability exploit, CVSS considers that a *scope change* has occurred. This condition is captured by the new metric, Scope.

As depicted in Figure 1, when scoring vulnerabilities in CVSS v3.0, the Exploitability metrics are scored relative to the vulnerable component. That is, they are scored by considering the component that suffers the coding flaw. On the other hand, the Impact metrics are scored relative to the impacted component. In some cases, the vulnerable component may be the same as the impacted component, in which case, no scope change has occurred. However, in other cases, there may be an impact to the vulnerable component, as well as to the impacted component. In these cases, a scope change has occurred, and the Confidentiality, Integrity and Availability Impact metrics should reflect the impact to either the vulnerable component, or the impacted component, whichever is most severe.

In the case of a vulnerability that allows the theft of a password file, while there may be subsequent steps the attacker takes to commit unauthorized account access, the most direct outcome is a loss of confidentiality of the local system file. As such, there would be no scope change. However, in the case of a vulnerability that allows a router's ARP table to be overwritten by an attacker, there are two impacts. First, to the router's system file (Integrity impact to the vulnerable component), and second, to those Internet services served by the router (Availability impact to affected systems). Because the score should reflect the most severe outcome, the impact

4 Note that while the vulnerable component will be a software program (host operating system, Internet application, device driver, etc,) the impacted component may be either another software program, a hardware device, or a network resource.

metric score may reflect either the Integrity loss to the vulnerable component, or the Availability loss to the Internet services, whichever is more severe.[5]

Access Vector

The Access Vector (from v2.0) has been renamed to Attack Vector, but still generally reflects the "remoteness" of the attacker relative to the vulnerable component. That is, the more remote an attacker is to the vulnerable component (in terms of logical and physical network distance), the greater the Base score will be. Further, this metric now distinguishes between local attacks which require local system access (such as with an attack against a desktop application) and physical attacks which require physical access to the platform in order to exploit a vulnerability (such as with a firewire, USB, or jailbreaking attack).

Attack Complexity

Access Complexity (from v2.0) conflated two issues: any software, hardware, or networking condition beyond the attacker's control that must exist or occur in order for the vulnerability to be successfully exploited (for example a software race condition, or application configuration), and the requirement for human interaction (for example, requiring a user execute a malicious executable). Therefore, Access Complexity has been separated into two metrics, Attack Complexity (which addresses the former condition), and **User Interaction** (which addresses the latter condition).

Privileges Required

The new metric, Privileges Required, replaces the Authentication metric of v2.0. Instead of measuring the *number of times* an attacker must separately authenticate to a system, Privileges Required captures the *level of access* required for a successful attack. Specifically, the metric values High, Low, and None reflect the privileges required by an attacker in order to exploit the vulnerability.

Impact Metrics

The Confidentiality, Integrity and Availability impact metric values from v2.0 of None, Partial, and Complete have been replaced with None, Low, and High. Rather than representing the overall *percentage* (proportion) of the systems impacted by an attack, the new metric values reflect the overall *degree* of impact caused by an attack. For example, while the Heartbleed[6] vulnerability only caused a loss to a small amount of information, the impact was quite severe. In CVSS v2.0, this would have been scored as Partial, while in CVSS v3.0, this is appropriately scored as High.

Additionally, in the example above, the impact metrics now reflect the consequence to the impacted component. And the impacted component may or may not be the same as the component that possesses the vulnerability being exploited.

Temporal Metrics

The influence of Temporal metrics has been reduced in v3.0, relative to v2.0. Exploitability has been renamed to Exploit Code Maturity to better represent what the metric is measuring.

---

5 See the Examples document which accompanies this guide for more information.
6 See http://heartbleed.com/.

Environmental Metrics

The Environmental metrics Target Distribution and Collateral Damage Potential have been replaced by Modified factors which accommodates mitigating controls or control weaknesses that may exist within the user's environment that could reduce or raise the impact of a successfully exploited vulnerability.

Qualitative Rating Scale

Some organizations created systems to map CVSS v2.0 Base scores to qualitative ratings. CVSS v3.0 now provides a standard mapping from numeric scores to the severity rating terms None, Low, Medium, High and Critical, as explained in the CVSS v3.0 specification document. The use of these qualitative severity ratings is optional, and there is no requirement to include them when publishing CVSS scores.

Organizations using CVSS v3.0 scores that wish to use an *alternate* severity rating system are asked to use different rating terms or to clearly state that their ratings do not comply with the CVSS v3.0 specification, to avoid confusion.

Summary of Changes

An important consequence of these changes is that v2.0 and v3.0 scores may not always be comparable. For example, a vulnerable application that could result in its complete compromise would have been scored in v2.0 with Confidentiality, Integrity and Availability impact metric values of Partial. Whereas in v3.0, this same vulnerability would now be scored with the equivalent Confidentiality, Integrity and Availability impact metric values of High.

A summary of changes from v2.0 are presented in Table 1.

Table 1: CVSS v2.0 to v3.0 Changes

Version 2.0	Version 3.0
Vulnerabilities are scored relative to the overall impact to the host platform.	Vulnerabilities now scored relative to the impact to the impacted component.
No awareness of situations in which a vulnerability in one application impacted other applications on the same system.	A new metric, Scope, now accommodates vulnerabilities where the *thing suffering the impact* (the impacted component) is different from *the thing that is vulnerable* (the vulnerable component).
Access Vector may conflate attacks that require local system access and physical hardware attacks.	Local and Physical values are now separated in the Attack Vector metric.
In some cases, Access Complexity conflated system configuration and user interaction.	This metric has been separated into Attack Complexity (accounting for system complexity), and User Interaction (accounting for user involvement in a successful attack).
In practice, the Authentication metric scores were biased toward two of three	A new metric, Privileges Required, replaces Authentication, and now reflects

possible outcomes, and not effectively capturing the intended aspect of a vulnerability.	the greatest privileges required by an attacker, rather than the number of times the attacker must authenticate.
Impact metrics reflected percentage of impact caused to a vulnerable application.	Impact metric values now reflect the degree of impact, and are renamed to None, Low and High.
The Environmental metrics of Target Distribution and Collateral Damage potential were not found to be useful.	Target Distribution and Collateral Damage potential have been replaced with Mitigating Factors.
CVSS v2.0 could not accommodate scoring multiple vulnerabilities used in the same attack.	While not a formal metric, guidance on scoring multiple vulnerabilities is provided with Vulnerability Chaining.
No formal qualitative scoring guidelines were provided.	Numerical ranges have been mapped to a 5-point qualitative rating scale.

Scoring Guide

Below are a number of recommendations for analysts when scoring vulnerabilities with CVSS v3.0.

CVSS Scoring in the Exploit Lifecycle

When understanding when to score the impact of vulnerabilities, analysts should constrain impacts to a reasonable final impact which they are confident an attacker is able to achieve. Ability to cause this impact should be supported by the Exploitability sub score as a minimum, but may also include details from the vulnerability's description. For example, consider the following two vulnerabilities:

In vulnerability 1, a remote, unauthenticated attacker can send a trivial, crafted request to a web server which causes the web server to disclose the plaintext password of the root (administrator) account. The analyst only knows from the Exploitability sub score metrics and the vulnerability description that the attacker has access to send a crafted request to the web server in order to exploit the vulnerability. Impact should stop there; while an attacker *may* be able to use these credentials to later execute code as the administrator, it is not known that the attacker has access to a login prompt or method to execute commands with those credentials. Gaining access to this password represents a direct, serious loss of Confidentiality only:

Base score: 7.5 [CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N].

In vulnerability 2, a local, low-privileged user can send a trivial, crafted request to the operating system which causes it to disclose the plaintext password of the root (administrator) account. The analyst knows from the Exploitability sub score metrics and the vulnerability description that the attacker has access to the operating system, and can log in as a local, low privileged attacker. Gaining access to this password represents a direct, serious loss of Confidentiality, Integrity, and Availability because the analyst can reasonably issue commands as the root / administrator account (assume that the attacker could log out from her own account and log back in as root):

Base score: 7.8 [CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H].

Confidentiality and Integrity, versus Availability Impacts

The Confidentiality and Integrity metrics refer to impacts that affect the *data* used by the service. For example, web content that has been maliciously altered, or system files that have been stolen. The Availability impact metric refers to the *operation* of the service. That is, the Availability metric speaks to the performance and operation of the service itself – not the availability of the data. Consider a vulnerability in an Internet service such as web, email, or DNS that allows an attacker to modify or delete all web files in a directory would incur an impact to Integrity only, rather than Availability. The reason is that the web service is still performing properly – it just happens to be serving back altered content.

Local vulnerabilities exploited by remote attackers

In CVSS v2.0, Scoring Tip 5 stated: "When a vulnerability can be exploited both locally and from the network, the Network value should be chosen. When a vulnerability can be exploited both locally and from adjacent networks, but not from remote networks, the Adjacent Network value should be chosen. When a vulnerability can be exploited from the adjacent network and remote networks, the Network value should be chosen." This guidance sometimes led to confusion in cases where an attacker would trick a user into downloading a malformed document from a remote web server, exploiting a file parsing vulnerability. In such case, analysts using CVSS v2.0 would treat these vulnerabilities as "network," producing scores with metric strings of: AV:N/AC:M/Au:N/C:P/I:P/A:P, or AV:N/AC:M/Au:N/C:C/I:C/A:C.

This guidance has been improved in CVSS v3.0 by clarifying the definitions of the Network and Adjacent values of the Attack Vector metric. Specifically, analysts should only score for Network or Adjacent when a vulnerability is bound to the network stack. Vulnerabilities which require user interaction to download or receive malicious content (which could also be delivered locally, e.g. via USB drives) should be scored as Local.

For example, a document parsing vulnerability, which does not rely on the network in order to be exploited, should typically be scored with the Local value, regardless of the method used to distribute such a malicious document (e.g. it could be a link to a web site, or via a USB stick).

Cross Site Scripting Vulnerabilities

In CVSS v2.0, specific guidance was necessary to produce non-zero scores for cross-site scripting (XSS) vulnerabilities, because vulnerabilities were scored relative to the host operating system that contained the vulnerability. A typical XSS vulnerability produced a score which described a partial integrity impact due to modification of the web server's response to the client: AV:N/AC:M/Au:N/C:N/I:P/A:N. This persisted even for DOM-based XSS vulnerabilities which, while they may be triggered by interaction with the server, are exploited entirely at the client-side (e.g. when server-delivered JavaScript parses the request string sent to the server).

This is one of the key scenarios for which Scope was designed – where impacts are suffered not by the vulnerable component (e.g. the web server, or the JavaScript delivered by the web server), but by a component whose privileges are managed by a separate authority (e.g. the client's browser environment). Therefore, under CVSS v3.0, cross-site scripting vulnerabilities do not have to be constrained to the limited or non-existent impacts to the server, and can now be scored for impacts that are realized at the client. A reflected XSS vulnerability that allowed an attacker to deliver a malicious link to a victim and execute JavaScript in their browser might be scored: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

Man in the Middle

CVSS v3.0 now explicitly accommodates scoring man-in-the-middle attacks. While not specifically addressed in v2.0, in v3.0, this type of attack is addressed with the Attack Complexity metric.

Hardware Vulnerabilities

In addition, while CVSS is primarily designed for scoring vulnerabilities and impacts to software, v3.0 is now better suited for also scoring impacts that include hardware components, and networking effects.

Vulnerability Chaining

CVSS is designed to classify and rate individual vulnerabilities. However, it is important to support the needs of the vulnerability analysis community by accommodating situations where multiple vulnerabilities are exploited in the course of a single attack to compromise a host or application. The scoring of multiple vulnerabilities in this manner is termed Vulnerability Chaining. Note that this is not a formal metric, but is included as guidance for analysts when scoring these kinds of attacks.

When scoring a chain of vulnerabilities, it is the responsibility of the analyst to identify which vulnerabilities are combined to form the chained score. The analyst should list the distinct vulnerabilities and their scores, along with the chained score. For example, this may be communicated within a vulnerability disclosure notice posted on a webpage.

In addition, the analyst may include other types of related vulnerabilities that could be chained with the vulnerabilities being scored. Specifically, the analyst may list generic types (or classes) of related vulnerabilities that are often chained together, or provide further descriptions of required preconditions that must exist. For example, one might describe how certain kinds of SQL Injection vulnerabilities are precursors to a cross-site scripting (XSS) attack, or how a particular kind of buffer overflow would grant local privileges. Listing the generic types or classes of vulnerabilities provides the minimum information necessary to warn other users, without potentially informing attackers about new exploit opportunities.

Alternatively, the analyst may identify (in the form of a machine readable and parseable list of vulnerabilities as CVE IDs or CWEs) a complete list of specific related vulnerabilities that are known to be (or are very likely to be) chained to one or more of the chained vulnerabilities being scored in order to exploit an IT system. In the event that a vulnerability can be exploited only after other preconditions are met (such as first exploiting another vulnerability), it is acceptable to combine two or more CVSS scores to describe the chain of vulnerabilities by scoring for the least-restrictive Exploitability sub score metrics and scoring for the most-impactful Impact sub score metrics. The following example uses the Exploitability, Scope, and Impact sub scores to describe the chain:

Vulnerability A is: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H, and as can be seen from the vector, requires a local, low-privileged user in order to exploit. Whereas Vulnerability B is, AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L which provides an unprivileged, remote attacker the ability to execute code on a system with Low impacts if a local user interacts to complete the attack. Therefore, given both A & B, Chain C could be described as the chain of B -> A: AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H which combines the Exploitability of B, the scope is unchanged in both cases, and the Impact of A, because if one can exploit B and gain the code

execution as a local user from it, then one has satisfied the prerequisite to subsequently launch A causing an impact from vulnerability A.

Glossary of Terms

Authority: A computing container that grants and manages privileges to resources. Examples of authorities include, a database application, an operating system, and a sandbox environment.

Chained Score: The Base score produced by scoring two or more chained vulnerabilities.

Chained Vulnerabilities: See Vulnerability Chaining.

Component: Refers to either a software or hardware component.

Software Component: A software program or module that contains computer instructions to be executed. E.g. an operating system, Internet application, device driver.

Hardware Component: A physical computing device.

Impacted Component: The component (or components) that suffer(s) the consequence of the exploited vulnerability. This (they) can either be the same component as the vulnerable component, or, if a scope changed has occurred, a different one.

Privileges: A collection of rights (typically read, write and execute) granted to a user or user process which defines access to computing resources.

Resources: A software or network object that is accessed, modified, or consumed by a computing device. E.g. computer files, memory, CPU cycles, or network bandwidth.

Scope: The collection of privileges defined and managed by an authorization authority when granting access to computing resources.

Vulnerability: A weakness or flaw in a software (or hardware) component.

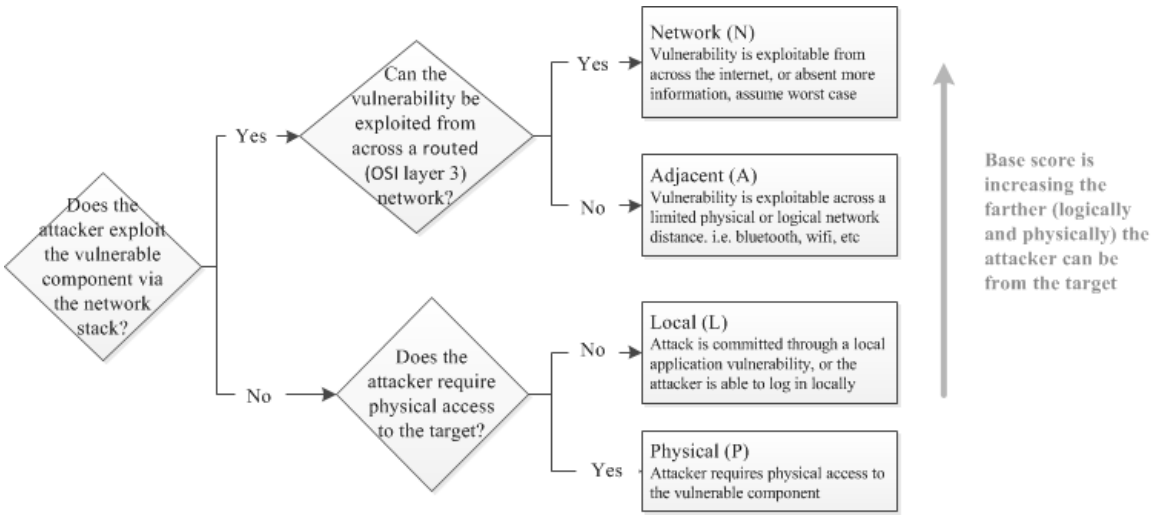
Vulnerability Chaining: The sequential exploit of multiple vulnerabilities in order to attack an IT system, where one or more exploits at the end of the chain require the successful completion of prior exploits in order to be exploited. See also the definition available at http://cwe.mitre.org/documents/glossary/#Chain.

Vulnerable component: The software (or hardware) component that bears the vulnerability, and that which would be patched.

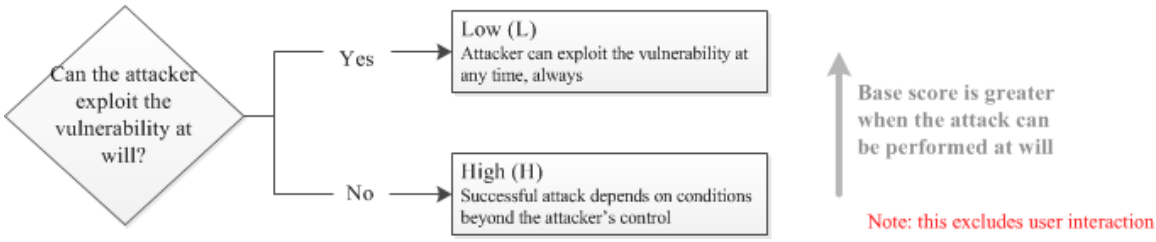
Scoring Rubric

The scoring rubric provides a quick reference to scoring vulnerabilities in v3.0. It is meant to supplement existing scoring discussion found in the Specification Document.

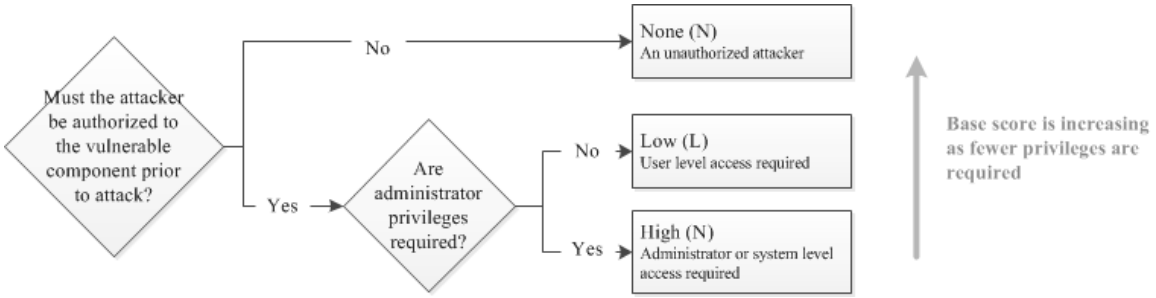
Attack Vector



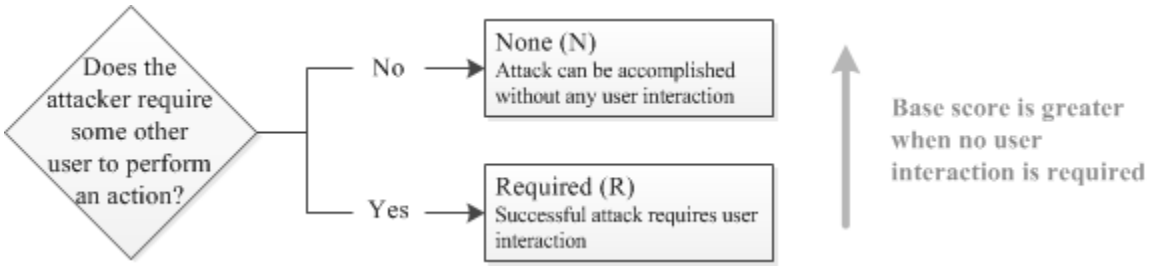
Attack Complexity



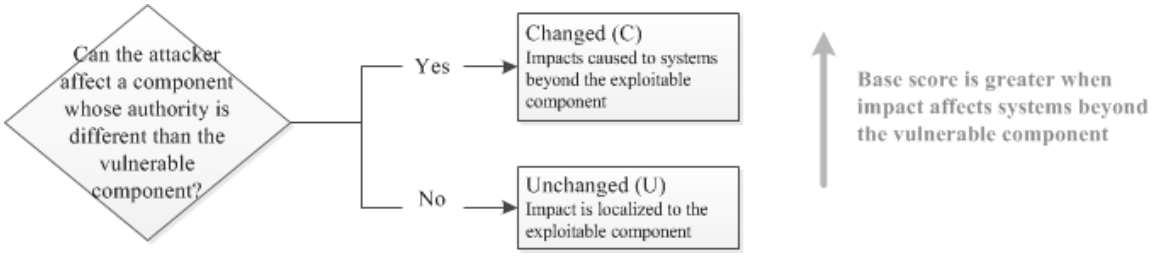
Privileges Required



User Interaction

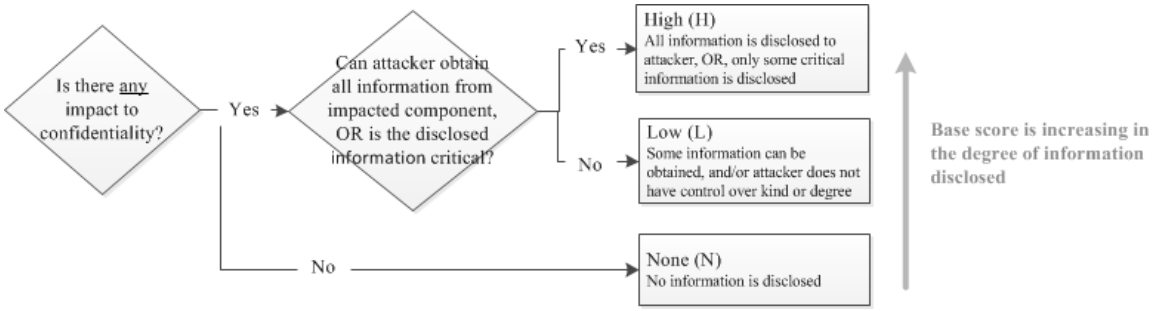


Scope

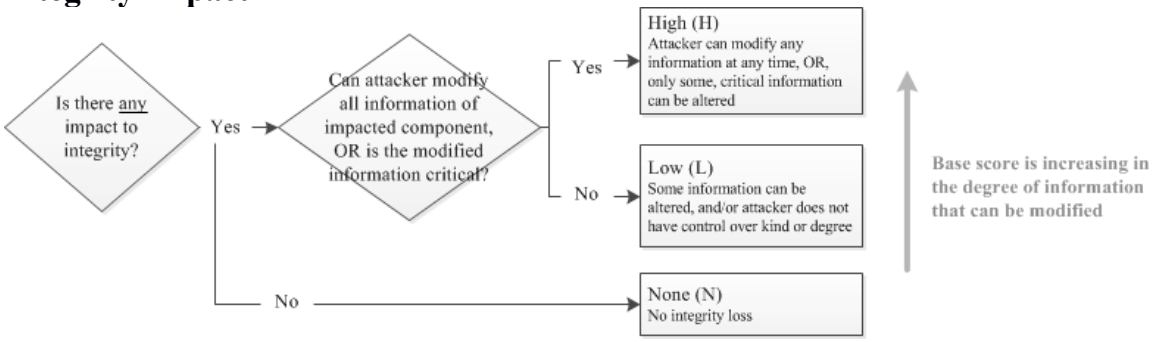


Note, if Scope change has not occurred, Confidentiality, Integrity and Availability impacts reflect consequence to the vulnerable component, otherwise they reflect consequence to the component that suffers the greater impact.

Confidentiality Impact



Integrity Impact



Availability Impact