2019 CWE Top 25 Most Dangerous Software Errors

Top 25 | Methodology | Scoring Metrics | On the Cusp | Limitations | Remapping | What Changed

Introduction

The Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Errors (CWE Top 25) is a demonstrative list of the most widespread and critical weaknesses that can lead to serious vulnerabilities in software. These weaknesses are often easy to find and exploit. They are dangerous because they will frequently allow adversaries to completely take over execution of software, steal data, or prevent the software from working. The CWE Top 25 is a community resource that can be used by software developers, software testers, software customers, software project managers, security researchers, and educators to provide insight into some of the most prevalent security threats in the software industry.

To create the list, the CWE Team used a data-driven approach that leverages published Common Vulnerabilities and Exposures (CVE®) data and related CWE mappings found within the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD), as well as the Common Vulnerability Scoring System (CVSS) scores associated with each of the CVEs. A scoring formula was then applied to determine the level of prevalence and danger each weakness presents. This data-driven approach can be used as a repeatable, scripted process to generate a CWE Top 25 list on a regular basis with minimal effort.

The CWE Top 25

Below is a brief listing of the weaknesses in the 2019 CWE Top 25, including the overall score of each.

Rank	ID	Name	Score
[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10
[11]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.47
[12]	CWE-787	Out-of-bounds Write	11.08
[13]	CWE-287	Improper Authentication	10.78
[14]	CWE-476	NULL Pointer Dereference	9.74
[15]	CWE-732	Incorrect Permission Assignment for Critical Resource	6.33

Rank	ID	Name	Score
[16]	CWE-434	Unrestricted Upload of File with Dangerous Type	5.50
[17]	CWE-611	Improper Restriction of XML External Entity Reference	5.48
[18]	CWE-94	Improper Control of Generation of Code ('Code Injection')	5.36
[19]	CWE-798	Use of Hard-coded Credentials	5.12
[20]	CWE-400	Uncontrolled Resource Consumption	5.04
[21]	CWE-772	Missing Release of Resource after Effective Lifetime	5.04
[22]	CWE-426	Untrusted Search Path	4.40
[23]	CWE-502	Deserialization of Untrusted Data	4.30
[24]	CWE-269	Improper Privilege Management	4.23
[25]	CWE-295	Improper Certificate Validation	4.06

Methodology

The 2019 CWE Top 25 was developed by obtaining published CVE vulnerability data found within the NVD. The NVD obtains vulnerability data from CVE and then supplements this data with additional analysis and data to provide more information about vulnerabilities. In addition to providing the underlying weakness for each vulnerability, the NVD provides a CVSS score, which is a numerical score representing the potential severity of a vulnerability based upon a standardized set of characteristics about the vulnerability. NVD provides this information in a digestible format that helps drive the data driven approach in creating the CWE Top 25. This approach provides an objective look at what vulnerabilities are currently seen in the real world, creates a foundation built on publicly reported vulnerabilities instead of relying on surveys and opinions, and makes the process repeatable in future years.

The 2019 CWE Top 25 leverages NVD data from the years 2017 and 2018, which consisted of approximately twenty-five thousand CVEs. The CWE team developed a scoring formula to calculate a rank order of weaknesses. The scoring formula combines the frequency that a CWE is the root cause of a vulnerability with the projected severity of its exploitation. In both cases, the frequency and severity are normalized relative to the minimum and maximum values seen.

To determine a CWE's frequency, the scoring formula calculates the number of times a CWE is mapped to a CVE within NVD. Only those CVEs that have an associated weakness are used in this calculation, since using the entire set of CVEs within NVD would result in very low frequency rates and very little difference amongst the different weakness types.

Freq = {count(CWE_X' ∈ NVD) for each CWE_X' in NVD}

Fr(CWE_X) = (count(CWE_X ∈ NVD) - min(Freq)) / (max(Freq) - min(Freq))

The other component in the scoring formula is a weakness' severity, which is represented by the average CVSS score of all CVEs that map to the particular CWE. The equation below is used to calculate this value.

Sv(CWE_X) = (average_CVSS_for_CWE_X - min(CVSS)) / (max(CVSS) - min(CVSS))

The level of danger presented by a particular CWE is then determined by multiplying the severity score by the frequency score.

Score(CWE_X) = Fr(CWE_X) * Sv(CWE_X) * 100

There are a few properties of the scoring method that merit further explanation.

- Weaknesses that are rarely exploited will not receive a high score, regardless of the typical severity associated with any exploitation. This makes sense, since if developers are not making a particular mistake, then the weakness should not be highlighted in the CWE Top 25.
- Weaknesses with a low impact will not receive a high score. This again makes sense, since the inability to cause significant harm by exploiting a weakness means that weakness should be ranked below those that can.
- Weaknesses that are both common and can cause harm should receive a high score.

The CWE Top 25 with Scoring Metrics

The following table shows the 2019 CWE Top 25 with relevant scoring information, including the number of entries related to a particular CWE within the NVD data set, and the average CVSS score for each weakness.

Rank	ID	NVD Count	Avg CVSS	Overall Score
[1]	CWE-119	3545	8.045	75.56
[2]	CWE-79	3430	5.778	45.69
[3]	CWE-20	2360	7.242	43.61
[4]	CWE-200	2300	5.961	32.12
[5]	CWE-125	1428	7.270	26.53
[6]	CWE-89	977	9.129	24.54
[7]	CWE-416	799	8.374	17.94
[8]	CWE-190	867	7.679	17.35
[9]	CWE-352	693	8.365	15.54
[10]	CWE-22	759	7.275	14.10
[11]	CWE-78	486	8.707	11.47
[12]	CWE-787	510	8.169	11.08
[13]	CWE-287	495	8.188	10.78
[14]	CWE-476	572	6.834	9.74
[15]	CWE-732	334	7.393	6.33
[16]	CWE-434	239	8.549	5.50
[17]	CWE-611	262	7.949	5.48
[18]	CWE-94	230	8.637	5.36
[19]	CWE-798	215	8.782	5.12
[20]	CWE-400	288	6.980	5.04
[21]	CWE-772	304	6.714	5.04
[22]	CWE-426	215	7.823	4.40
[23]	CWE-502	177	8.921	4.30
[24]	CWE-269	226	7.332	4.23
[25]	CWE-295	248	6.658	4.06

Weaknesses On the Cusp

As in years past, the CWE team feels it is important to share additional CWEs that scored just outside of the top 25. Per the 2019 scoring formula against the NVD dataset, these weaknesses were potentially not severe enough, or not prevalent enough, to be included in the 2019 list.

Developers that complete mitigation and risk decision-making on the 2019 CWE Top 25 may want to look for these other weaknesses potentially present in their software. For these reasons, users of the 2019 CWE Top 25 should seriously consider including these additional weaknesses in their analyses:

Rank	ID	Name	NVD Count	Avg CVSS	Overall Score
[26]	CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')	218	6.610	3.53
[27]	CWE-522	Insufficiently Protected Credentials	150	8.460	3.39
[28]	CWE-704	Incorrect Type Conversion or Cast	143	8.484	3.25

Rank	ID	Name	NVD Count	Avg CVSS	Overall Score
[29]	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	187	6.740	3.11
[30]	CWE-918	Server-Side Request Forgery (SSRF)	128	7.917	2.65
[31]	CWE-415	Double Free	111	7.981	2.32
[32]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')	159	6.141	2.31
[33]	CWE-863	Incorrect Authorization	113	7.050	2.00
[34]	CWE-862	Missing Authorization	92	7.491	1.76
[35]	CWE-532	Inclusion of Sensitive Information in Log Files	90	7.064	1.59
[36]	CWE-306	Missing Authentication for Critical Function	66	8.529	1.50
[37]	CWE-384	Session Fixation	76	7.083	1.34
[38]	CWE-326	Inadequate Encryption Strength	73	7.278	1.34
[39]	CWE-770	Allocation of Resources Without Limits or Throttling	75	6.880	1.27
[40]	CWE-617	Reachable Assertion	75	6.729	1.23

Limitations of the Methodology

There are several limitations to the data-driven approach chosen by the CWE Team.

Data Bias

First, the approach only uses data that was publicly reported and captured in NVD, and numerous vulnerabilities exist that do not have CVE IDs. Vulnerabilities that are not included in NVD are therefore excluded from this approach. For example, CVE/NVD typically does not cover vulnerabilities found and fixed before any software has been publicly released, in online services, or in bespoke software that is internal to a single organization. Weaknesses that lead to these types of vulnerabilities may be under-represented in the 2019 CWE Top 25.

Second, even for vulnerabilities that receive a CVE, often there is not enough information to make an accurate (or precise) identification of the appropriate CWE that is exploited. Many CVE entries are published by software vendors who only describe the impact of the vulnerability without providing details of the vulnerability itself. In other cases, the CVE description covers how the entry is attacked – but this does not always indicate what the associated vulnerability is. For example, if a long input to a program causes a crash, the cause of the crash could be due to a buffer overflow, a reachable assertion, excessive memory allocation, an unhandled exception, etc. In other CVE entries, only generic terms are used such as "malicious input," which gives no indication of the associated weakness. For some entries, there may be useful information available in the references, but it is difficult to analyze. For example, a researcher might use a fuzzing program that generates a useful test case that causes a crash, but the developer simply fixes the crash without classifying and reporting what the underlying mistake was. The CWE Team identified at least 2,600 CVEs that had insufficient details to perform a precise CWE mapping.

Third, there is inherent bias in the CVE/NVD dataset due to the set of vendors that report vulnerabilities and the languages that are used by those vendors. If one of the largest contributors to CVE/NVD primarily uses C as its programming language, the weaknesses that often exist in C programs are more likely to appear. Fuzzing programs can be very effective against memory-based programs, so they may find many more vulnerabilities. The scoring metric outlined above attempts to mitigate this bias by looking at more than just the most frequently reported CWEs; it also takes into consideration average CVSS score.

Another bias in the CVE/NVD dataset is that most vulnerability researchers and/or detection tools are very proficient at finding certain weaknesses but do not find other types of weaknesses. Those types of weakness that researchers and tools struggle to find will end up being under-represented within the 2019 CWE-Top 25.

Finally, gaps or perceived mischaracterizations of the CWE hierarchy itself lead to incorrect mappings. The ongoing remapping work helps the CWE Team learn about these content gaps and issues, which will be addressed in subsequent CWE releases.

Metric Bias

An important bias to understand related to the metric is that it indirectly prioritizes implementation flaws over design flaws, due to their prevalence within individual software packages. For example, a web application may have many different cross-site scripting (XSS) vulnerabilities due to large attack surface, yet only one instance of use of an insecure cryptographic algorithm.

Remapping Task

To prepare the CVE/NVD data for analysis, the CWE Team reviewed the CWE mappings of selected CVE/NVD entries and, where appropriate, "remap" the entries so that they reference more appropriate CWE IDs.

For remapping, the CWE team used a broader range of CWE IDs than NVD analysts previously had available, and the team was stricter about explicitly labeling insufficient information. This remapping work was performed on thousands of CVE entries. It produced a richer list of CWE IDs for NVD analysts to choose from, as reflected in the updated CWE-1003 view. The remapped data has been shared with NIST so that they can update their CVE entries within NVD.

The remapping was performed iteratively, and the team eventually developed a more repeatable process that can be performed in future versions of the Top 25. The primary activities were:

- Remap all CVE entries that were mapped to CWE categories. While categories such as CWE-399 (Resource Management Errors) were available to NVD analysts for many years as part of view CWE-1003, the CWE Project now advises that categories should only serve as organizational and navigational mechanisms that should not be mapped; only weaknesses should be used.
- Perform automated keyword searches to find likely remaps to CWE entries that were not on NVD analysts' CWE lists. For example, out-of-bounds read (CWE-125) was not in the original NVD mapping lists, and the associated CVE entries were mapped to the higher-level CWE-119. However, phrases related to out-of-bounds read were easily findable within many CVE descriptions.
- Focus on high-level CWE classes that might have more precise mappings. This work was labor-intensive, and the CWE team was unable to cover the all class-level entries due to the large number of CVE entries that were mapped to them, such as CWE-119, CWE-20, and CWE-200. However, the effort yielded significant improvements to NVD data, the CWE-1003 view, and a better understanding of the needs for reorganization in some parts of the CWE hierarchy.
- Identify specific CVE entries that might indicate gaps within CWE itself, that is, CVE entries that have sufficient technical details to understand the weakness, but the weakness does not have an appropriate CWE to be mapped.
- Update the CWE-1003 view to remove the categories; ensure there is coverage with the most common CWE mappings; and to change from a deep structure to a hierarchy with only two levels, generally with a class as a parent and bases as children.

What Changed

The major difference between the 2011 and 2019 CWE Top 25 lists is in the approach that was used. The 2011 CWE/SANS Top 25 was constructed using surveys and personal interviews with developers, top security analysts, researchers, and vendors. These responses were normalized based on the prevalence and ranked by the CWSS methodology. The 2019 CWE Top 25, on the other hand, was formed based on real-world vulnerabilities found in the NVD.

As a result of this change in approach, a few changes to the list of weaknesses that make up the CWE Top 25 was expected. The biggest of these changes is the inclusion of some class level CWEs

that represent broad types of errors: CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer), CWE-20 (Improper Input Validation), CWE-200 (Information Exposure) and CWE-287 (Improper Authentication). These entries are often used by vendors and researchers to describe the root cause of a vulnerability being reported and are therefore prevalent in our data-driven analysis of NVD. Looking closer, however, these high-level weaknesses are often the parent of more detailed weaknesses that appeared in previous Top 25 lists. For example, CWE-119 is the parent of CWE-120. While the latter was #3 in the 2011 list, it is not found on the 2019 list despite the former being the new #1. Similarly, CWE-287 is #13 in 2019 but does not at appear in 2011. Looking closer however shows that CWE-287 is the parent of CWE-306 (#5 on the 2011 list), CWE-862 (#6 on the 2011 list), and CWE-863 (#15 on the 2011 list), none of which are found on the 2019 list.

Another interesting change is that some weaknesses in the 2019 list are reported at a different place within a potential chain of weaknesses. For example, CWE-787 (Out-of-bounds Write) did not appear in the 2011 list but is #12 in 2019. CWE-787 is often part of a chain that starts with CWE-120, which was #3 in 2011.

A few other 2019 CWE Top 25 List entries also deserve some attention.

- CWE-125 (Out-of-bounds Read) appeared much higher in the list than expected (#5). The CWE team believes this might be due to increased instances of pointing to this entry for complex exploit chains, kernel elevation of privilege, and improved detection methods in the aftermath of Heartbleed (whose discovery revealed imperfections in static code analysis techniques)
- CWE-417 (Use After-Free), CWE-611 (Improper Restriction of XML External Entity Reference), and CWE-502 (Deserialization of Untrusted Data) appear at #7, #17, and #23 respectively, but were not present at all in 2011. The CWE team believes this is probably a reflection of an increase in exploitation capability
- CWE-476 (NULL Pointer Dereference) appears at #14 and not at all in the 2011 Top 25. The CWE review team believe this is likely due to the increased use of fuzzing

Lastly, CWE-20 and CWE-200 (#3 and #4, respectively), are class level weaknesses and well-known secure coding problem areas. The CWE team believes there are potentially instances when these entries are used for mapping vulnerabilities to CWE when more specific, lower-level weakness types might be more appropriate. This is an area that the CWE Team is further investigating and hopes to improve in future versions of the CWE Top 25.

Archive

Past versions of the CWE Top 25 are available in the Archive.