2020 CWE Top 25 Most Dangerous Software Weaknesses

cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html

The 2020 Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Weaknesses (CWE Top 25) is a demonstrative list of the most common and impactful issues experienced over the previous two calendar years. These weaknesses are dangerous because they are often easy to find, exploit, and can allow adversaries to completely take over a system, steal data, or prevent an application from working. The CWE Top 25 is a valuable community resource that can help developers, testers, and users — as well as project managers, security researchers, and educators — provide insight into the most severe and current security weaknesses.

To create the 2020 list, the CWE Team leveraged Common Vulnerabilities and Exposures (CVE®) data 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 CVE. A formula was applied to the data to score each weakness based on prevalence and severity.

The CWE Top 25

Below is a brief listing of the weaknesses in the 2020 CWE Top 25, including the overall score of each.

Rank	ID	Name	Score
[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.82
[2]	CWE-787	Out-of-bounds Write	46.17
[3]	CWE-20	Improper Input Validation	33.47
[4]	CWE-125	Out-of-bounds Read	26.50
[5]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	23.73
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	20.69
[7]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	19.16
[8]	CWE-416	Use After Free	18.87

Rank	ID	Name	Score
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	17.29
[10]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	16.44
[11]	CWE-190	Integer Overflow or Wraparound	15.81
[12]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	13.67
[13]	CWE-476	NULL Pointer Dereference	8.35
[14]	CWE-287	Improper Authentication	8.17
[15]	CWE-434	Unrestricted Upload of File with Dangerous Type	7.38
[16]	CWE-732	Incorrect Permission Assignment for Critical Resource	6.95
[17]	CWE-94	Improper Control of Generation of Code ('Code Injection')	6.53
[18]	CWE-522	Insufficiently Protected Credentials	5.49
[19]	CWE-611	Improper Restriction of XML External Entity Reference	5.33
[20]	CWE-798	Use of Hard-coded Credentials	5.19
[21]	CWE-502	Deserialization of Untrusted Data	4.93
[22]	CWE-269	Improper Privilege Management	4.87
[23]	CWE-400	Uncontrolled Resource Consumption	4.14
[24]	CWE-306	Missing Authentication for Critical Function	3.85
[25]	CWE-862	Missing Authorization	3.77

Back to top

Analysis and Comment

The major difference between the 2019 and 2020 CWE Top 25 lists is the increased transition to more specific weaknesses as opposed to abstract class-level weaknesses. While these class-level weaknesses still exist in the list, they have moved down in the ranking. This movement is expected to continue in future years as the community improves its mapping to more specific weaknesses. Looking at the list, class-level weaknesses CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer), CWE-20 (Improper Input Validation), and CWE-200 (Exposure of Sensitive Information to an Unauthorized Actor) each move down a couple of spots; while more specific weaknesses like CWE-79 (Improper Neutralization of Input During Web Page Generation), CWE-787 (Out-of-bounds Write) and CWE-125 (Out-of-bounds Read) moved up to take their place. This change, and subsequent future movement, will greatly benefit users that are attempting to understand the actual issues that threaten today's systems.

The biggest movement up the list involves four weaknesses that are related to Authentication and Authorization:

- CWE-522 (Insufficiently Protected Credentials): from #27 to #18
- CWE-306 (Missing Authentication for Critical Function): from #36 to #24
- CWE-862 (Missing Authorization): from #34 to #25
- CWE-863 (Incorrect Authorization): from #33 to #29

All four of these weaknesses represent some of the most difficult areas to analyze a system on. A theory about this movement is that the community has improved its education, tooling, and analysis capabilities related to some of the more implementation specific weaknesses identified in previous editions of the CWE Top 25 and have reduced the occurrence of those, thus lowering their ranking, and in turn raising the ranking of these more difficult weaknesses. Four of the biggest movers down are:

- CWE-426 (Untrusted Search Path): from #22 to #26
- CWE-295 (Improper Certificate Validation): from #25 to #28
- CWE-835 (Loop with Unreachable Exit Condition): from #26 to #36
- CWE-704 (Incorrect Type Conversion or Cast): from #28 to #37

Another big movement is again the result of mapping to a more specific weakness. In 2019, CWE-772 (Missing Release of Resource after Effective Lifetime) was #21. However, this didn't tell the entire story as which type of resource not being released is important. For 2020, a more specific mapping was used to show the exact type of resource. Due to this change, CWE-401 (Missing Release of Memory after Effective Lifetime) went from not being on the list to being #32, and CWE-772 representing all non-memory resources dropped to #75. This change creates a more accurate CWE Top 25 and identifies the actual issue more precisely.

Also of note is the addition of CWE-77 (Improper Neutralization of Special Elements used in a Command) to the On the Cusp list at #31. "Command Injection" is a term that is used inconsistently in vulnerability descriptions, often at the expense of the more accurate "OS

Command Injection," or used to describe the resulting consequence and not the root weakness. This is an area of research for the CWE Team and one that will hopefully be improved in future releases of the CWE Top 25.

Methodology

The 2020 CWE Top 25 was developed by obtaining published vulnerability data from the NVD. The NVD obtains vulnerability data from CVE and then supplements it with additional analysis and information including a mapping to one or more weaknesses, and 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 2020 CWE Top 25. This approach provides an objective look at what vulnerabilities are currently seen in the real world, creates a foundation of analytical rigor built on publicly reported vulnerabilities instead of subjective surveys and opinions, and makes the process easily repeatable.

The 2020 CWE Top 25 leverages NVD data from the years 2018 and 2019, which consists of approximately 27,000 CVEs that are associated with a weakness. A scoring formula is used to calculate a ranked order of weaknesses which 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 the NVD. Only those CVEs that have an associated weakness are used in this calculation, since using the entire set of CVEs within the 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 methodology that merit further explanation.

- Weaknesses that are rarely discovered will not receive a high score, regardless of the typical consequence 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 significant harm should receive a high score.

Back to top

The CWE Top 25 with Scoring Metrics

The following table shows the 2020 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	CWE	NVD Count	Avg CVSS	Overall Score
[1]	CWE-79	3788	5.80	46.82
[2]	CWE-787	2225	8.31	46.17
[3]	CWE-20	1910	7.35	33.47
[4]	CWE-125	1578	7.13	26.5
[5]	CWE-119	1189	8.08	23.73
[6]	CWE-89	901	8.98	20.69
[7]	CWE-200	1467	6.01	19.16
[8]	CWE-416	918	8.26	18.87
[9]	CWE-352	866	8.08	17.29
[10]	CWE-78	767	8.52	16.44
[11]	CWE-190	846	7.70	15.81
[12]	CWE-22	792	7.27	13.67
[13]	CWE-476	529	6.83	8.35
[14]	CWE-287	412	8.05	8.17
[15]	CWE-434	346	8.50	7.38
[16]	CWE-732	426	6.99	6.95
[17]	CWE-94	295	8.74	6.53

Rank	CWE	NVD Count	Avg CVSS	Overall Score
[18]	CWE-522	283	7.92	5.49
[19]	CWE-611	277	7.88	5.33
[20]	CWE-798	234	8.76	5.19
[21]	CWE-502	217	8.93	4.93
[22]	CWE-269	278	7.36	4.87
[23]	CWE-400	249	7.09	4.14
[24]	CWE-306	193	8.10	3.85
[25]	CWE-862	236	6.90	3.77

Back to top

Weaknesses On the Cusp

Continuing on the theme from last year, the CWE team feels it is important to share these fifteen additional weaknesses that scored just outside of the final Top 25. Per the scoring formula, these weaknesses were potentially not severe enough, or not prevalent enough, to be included in the 2020 CWE Top 25.

Individuals that perform mitigation and risk decision-making using the 2020 CWE Top 25 may want to consider including these additional weaknesses in their analyses:

Rank	CWE	Name	NVD Count	Avg CVSS	Overall Score
[26]	CWE-426	Untrusted Search Path	175	7.68	3.25
[27]	CWE-918	Server-Side Request Forgery (SSRF)	161	7.85	3.08
[28]	CWE-295	Improper Certificate Validation	180	7.19	3.04
[29]	CWE-863	Incorrect Authorization	189	6.82	2.97
[30]	CWE-284	Improper Access Control	173	7.22	2.94
[31]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	131	8.46	2.77
[32]	CWE-401	Missing Release of Memory after Effective Lifetime	189	6.43	2.72

Rank	CWE	Name	NVD Count	Avg CVSS	Overall Score
[33]	CWE-532	Insertion of Sensitive Information into Log File	154	6.82	2.42
[34]	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	157	6.68	2.39
[35]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')	176	6.12	2.35
[36]	CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')	150	6.72	2.30
[37]	CWE-704	Incorrect Type Conversion or Cast	109	8.48	2.30
[38]	CWE-415	Double Free	117	8.04	2.30
[39]	CWE-770	Allocation of Resources Without Limits or Throttling	139	7.06	2.29
[40]	CWE-59	Improper Link Resolution Before File Access ('Link Following')	122	7.07	2.01

Back to top

Limitations of the Methodology

There are several limitations to the data-driven approach used in creating the CWE Top 25.

Data Bias

First, the approach only uses data that was publicly reported and captured in the NVD, and numerous vulnerabilities exist that do not have CVE IDs. Vulnerabilities that are not included in the NVD are therefore excluded from this approach. For example, CVE/NVD typically does not cover vulnerabilities found and fixed before any system 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 2020 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 being exploited. Many CVE entries are published by vendors who only describe the impact of the vulnerability without providing details of the vulnerability itself. For example, over 3,200 CVEs from 2018 and 2019 did not have sufficient information to determine the underlying weakness. In other cases, the CVE description covers how the vulnerability is attacked – but this does not always indicate what the associated weakness 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. These all correspond to different, individual CWEs. 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.

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 not others. Those types of weakness that researchers and tools struggle to find will end up being under-represented within the 2020 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.

Back to top

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, “re-mapped” the entries so that they reference more appropriate CWE IDs.

This re-mapping work was performed on over ten thousand CVE entries in consideration for the 2020 Top 25 List. The remapped data has been shared with NIST so that they can update their CVE entries within NVD.

The primary activities were:

- Remap 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, recent guidance and changes to view CWE-1003 have attempted to eliminate this practice. The CWE Team advises that categories should only serve as organizational and navigational mechanisms that should not be mapped to; only weaknesses should be used.
- Perform automated keyword searches to find likely remaps to CWE entries that were incorrectly mapped. For example, some CVE entries were mapped to the higher-level CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer). However, phrases related to out-of-bounds read were automatically discoverable within CVE descriptions. In those cases, mapping to the lower level CWE-125 (Out-of-bounds Read) is considered more appropriate.
- Focus on high-level CWE classes that might have more precise mappings. This work was labor-intensive, involving investigation into detailed references such as open source bug reports or security researcher advisories. 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, e.g., CWE-119, CWE-20, CWE-200, CWE-74, and CWE-284. This will be a continued focus in future years.
- 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 ensure there is coverage with the most common CWE mappings; while continuing to support a simplified structure with only two levels, generally with a class as a parent and bases as children. Four weaknesses were added to the CWE-1003 view based on frequent use in mapping: CWE-77, CWE-401, CWE-917, and CWE-1236.
- While the CWE team made every possible effort to minimize subjectivity in the remapping corrections, the lack of relevant, detailed information present in some CVE descriptions meant that a small portion of the dataset still required some subjective analytical conclusions.

Back to top

Archive

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