CWE Top 25 Most Dangerous Software Weaknesses, 2020

CWE Top 25 Most Dangerous Software Weaknesses, 2020

Cybernetic GI Security Bulletin provides a summary of CWE Top 25 Most Dangerous Software Weaknesses in the 2020. Entries may include additional information provided by organizations and efforts sponsored by Cybernetic GI. This data may include identifying information, values, definitions, and related links. The patch information is provided to users when available. Please note that some of the information in the bulletin is compiled from external, open source reports and is not a direct result of Cybernetic GI analysis .

The Homeland Security Systems Engineering and Development Institute, sponsored by the Department of Homeland Security and operated by MITRE, has released the 2020 Common Weakness Enumeration (CWE) Top 25 Most Dangerous Software Weaknesses list. The Top 25 uses data from the National Vulnerability Database (NVD) to compile the most frequent and critical errors that can lead to serious vulnerabilities in software. An attacker can often exploit these vulnerabilities to take control of an affected system, obtain sensitive information, or cause a denial-of-service condition.

Cybernetic Global Intelligence encourages users and administrators to review the Top 25 list and evaluate recommended mitigations to determine those most suitable to adopt.

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, 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.

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')	46.8
[2]	CWE-787	Out-of-bounds Write	46.2
[3]	CWE-20	Improper Input Validation	33.5
[4]	CWE-125	Out-of-bounds Read	26.5
[5]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	23.7
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	20.7
[7]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	19.2
[8]	CWE-416	Use After Free	18.9
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	17.3
[10]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	16.4
[11]	CWE-190	Integer Overflow or Wraparound	15.8
[12]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	13.7
[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

1. CWE-79 Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’)

Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’) occurs when “software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.”

According to CWE-79, Cross-site scripting (also known as “XSS”) consists of three main types:

- Type 1:Reflected XSS (or Non-Persistent)
- Type 2: Stored XSS (or Persistent) also known as “HTML injection”
- Type 0: DOM-Based XSS.

An attacker could inject a malicious script (into as web app for example) and then perform variety of malicious activities such as

Once the malicious script is injected, the attacker can perform a variety of malicious activities, such as stealing private information, launch phishing attacks or exploit vulnerabilities in victim’s web browser.

2. CWE-787 Out-of-bounds Write

The Out-of-bounds Write occurs when software writes data past the end, or before the beginning, of the intended buffer, according to CVE.

As explained in more detail in CWE-787, this software weakness “can result in corruption of data, a crash, or code execution. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent write operation then produces undefined or unexpected results.”

3. CWE-20 Improper Input Validation

The CWE-20 error highlights discrepancies in the data flow of a program. Parts of the system may receive unintended input as your app fails to validate the information. Programs that receive external data are most vulnerable to this error. Attackers exploit this error with input that an app can’t interpret. Successful attacks may either lead to arbitrary code execution or altered data flow.

Alternatively, attackers can opt for malicious input that modifies existing data. Most of these attacks target confidential data. CWE-20 has a close association with CWE-116 that deals with improper encoding or escaping of output. Apps that aren’t vulnerable to such attacks often preserve the meaning of a structured message.

According to CWE-20, Improper input validation may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

4. CWE-125 Out-of-bounds Read

CWE-125 highlights its prevalence in different apps. In this error, buffers placed in a system do not control how much data the software reads. With extended reading capabilities, attackers easily exploit different memory locations and read sensitive information like memory addresses in the process.
In the worst-case scenario, your system may crash. Buffer overflows, or segmentation faults often occur when attackers exploit this vulnerability. C and C++ software run the highest risk for this error. With appropriate input validation measures, developers should be able to mitigate against this vulnerability.

5. CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

According to CWE-119, an Improper Restriction of Operations within the Bounds of a Memory Buffer weakness occurs when the software "performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer."
As a consequence, an attacker could then:

- exploit to execute arbitrary code
- alter the intended control flow
- read sensitive information
- or cause the system to crash.

These consequences depend on the chip architecture, platform, and programming language that you use for app development. Developers can mitigate this vulnerability using memory management support. Such support systems ensure that buffer systems extend to the intended memory locations.

6. CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

According to CWE-89, a SQL Injection weakness exists when the "software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component."
As a consequence, an attacker could then alter query logic to bypass security checks or even insert additional statements that modify the back-end database and execute system commands.

This issue is also most common in database-driven websites.

7. CWE-200 Exposure of Sensitive Information to an Unauthorized Actor

CWE-200 exposes information to unauthorized individuals. Primary information exposures entail things like cryptography timing discrepancies. Resultant exposures, on the other hand, could involve scripts that lay bare a program. The severity of these exposures often depends on the information in question.

Often times, this weakness can be caused by coding mistakes such as:

- Lack of scrubbing or sanitizing data (explicitly inserts)
- Web script errors that reveal the full system path of the program (indirectly inserts).
- Unintentionally made accessible to unauthorized actors (such as web resources open to public internet).

To add, “information disclosure” is also sometimes used as an an alternate term for this flaw or related vulnerabilities. The information could be like a private message or something that helps attackers exploit a program. Developers should strive to create ‘safe’ areas within their systems. With trust boundaries around such areas, designers can offer fewer system privileges and keep this vulnerability in check.

8. CWE-416 Use After Free

According to CWE-416, Use After Free weakness occurs when software “references memory after it has been freed can cause a program to crash, use unexpected values, or execute code.” The use of previously freed memory can have any number of adverse consequences ranging from the corruption of valid data to the execution of arbitrary code, depending on the instantiation and timing of the flaw.

Also known as “dangling pointer,” use after free errors can occur when:

- Error conditions and other exceptional circumstances.
- Confusion over which part of the program is responsible for freeing the memory.

9. CWE-352 Cross-Site Request Forgery (CSRF)

According to CWE-352, Cross-Site Request Forgery (CSRF) happens when a “web application does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request.”
When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in data disclosure or unintended code execution
CSRF is also known by terms XSRF, Cross Site Reference Forgery and Session Riding.

10. CWE-78: Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

According to CWE-78, ‘OS Command Injection’ occurs when the software “constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.”
OS Command Injection has also gone by the terms Shell injection and Shell metacharacters.

11. CWE-190: Integer Overflow or Wraparound

According to CWE-190, Integer Overflow or Wraparound exists when “software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.”

12. CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

According to CWE-22, 'Path Traversal' occurs when "software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory."
Attackers can use use special elements such as ".." and "/" separators to escape outside of the restricted location and gain access to sensitive files or directories hosted on the system.

Alternate terms include Directory traversal and Path traversal.

13. CWE-476: NULL Pointer Dereference

According to CWE-476, NULL Pointer Dereference exists when the "application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit."
Furthermore, this software weakness can be caused by race conditions and simple programming omission flaws.

14. CWE-287: Improper Authentication

According to CWE-287, Improper Authentication occurs when a bad actor "claims to have a given identity, the software does not prove or insufficiently proves that the claim is correct."
Alternative terms include "authentification" (often used in non-English speaking countries) and AuthC.

15. CWE-434: Unrestricted Upload of File with Dangerous Type

According to CWE-434, Unrestricted Upload of File with Dangerous Type happens when software "allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment."
"Unrestricted file upload" is also used as an alternate term, but can be confused with related but different resource consumption vulnerabilities that are caused by lack of restrictions on the size or number of uploaded files.

16. CWE-732: Incorrect Permission Assignment for Critical Resource

According to CWE-732, Incorrect Permission Assignment for Critical Resource exists when software "specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors."
To add, CWE warns this flaw is especially dangerous when the resource is related to program configuration, execution or sensitive user data.

17. CWE-94: Improper Control of Generation of Code ('Code Injection')

According to CWE-94, 'Code Injection' weakness occurs when the software "constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment."

To add, an attacker could craft code to alter the intended control flow of the software and also lead to arbitrary code execution.

18. CWE-522: Insufficiently Protected Credentials

According to CWE-522, Insufficiently Protected Credentials exists when the product "transmits or stores authentication credentials, but it uses an insecure method that is susceptible to unauthorized interception and/or retrieval."

19. CWE-611: Improper Restriction of XML External Entity Reference

According to CWE-611, Improper Restriction of XML External Entity Reference happens when software "processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output."
As a result, an attacker can submit an XML file that defines an external entity with a file:// URI, thus cause the processing application to read the contents of a local file.

20. CWE-798: Use of Hard-coded Credentials

According to CWE-798, Use of Hard-coded Credentials occurs when the software "contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data."
Attackers can exploit hard-coded credentials, a significant hole, to bypass authentication configured by software administrators.

21. CWE-502: Deserialization of Untrusted Data

According to CWE-502, this software flaw exists when an application "deserializes untrusted data without sufficiently verifying that the resulting data will be valid."
Alternate terms include Unmarshaling, Unpickling (Python functionality) and PHP Object Injection (PHP apps).

22. CWE-269: Improper Privilege Management

According to CWE-269, Improper Privilege Management occurs when software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor. The weakness is introduced during Architecture and Design, Implementation and Operation stages. This weakness is caused during implementation of an architectural security tactic.

23. CWE-400: Uncontrolled Resource Consumption

According to CWE-400, Uncontrolled Resource Consumption exists when software "does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources."
Bad actors could take advantage of limited resources, such as memory, file storage and CPU, if software does not control or limit the number or size of the resources. As a result, actors could cause a denial of service (DoS) that consumes all available resources and cause a system crash.

24. CWE-306: Missing Authentication for Critical Function

According to CWE-306, Missing Authentication for Critical Function happens when the software "does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources."
For example, attackers could exploit exposed critical functionality to gain the same privileged level of the functionality. As a consequence, they could read/modify sensitive data, access administrative functionality or even execute arbitrary code.

25. CWE-862: Missing Authorization

According to CWE-862, Missing Authorization occurs when the software "does not perform an authorization check when an actor attempts to access a resource or perform an action."
Also known as "AuthZ", Authorization is the process of determining whether that user can access a given resource, based on the user's privileges and access control permissions applicable to the resource.

As a consequence, AuthZ flaws can lead to information exposure, denial of service and arbitrary code execution.