Fortify Audit Workbench

CWE Top 25 2023

DADAGAS

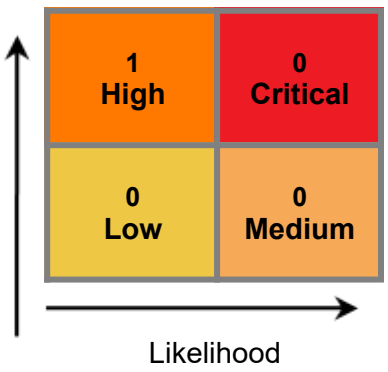
Table of Contents

Executive Summary

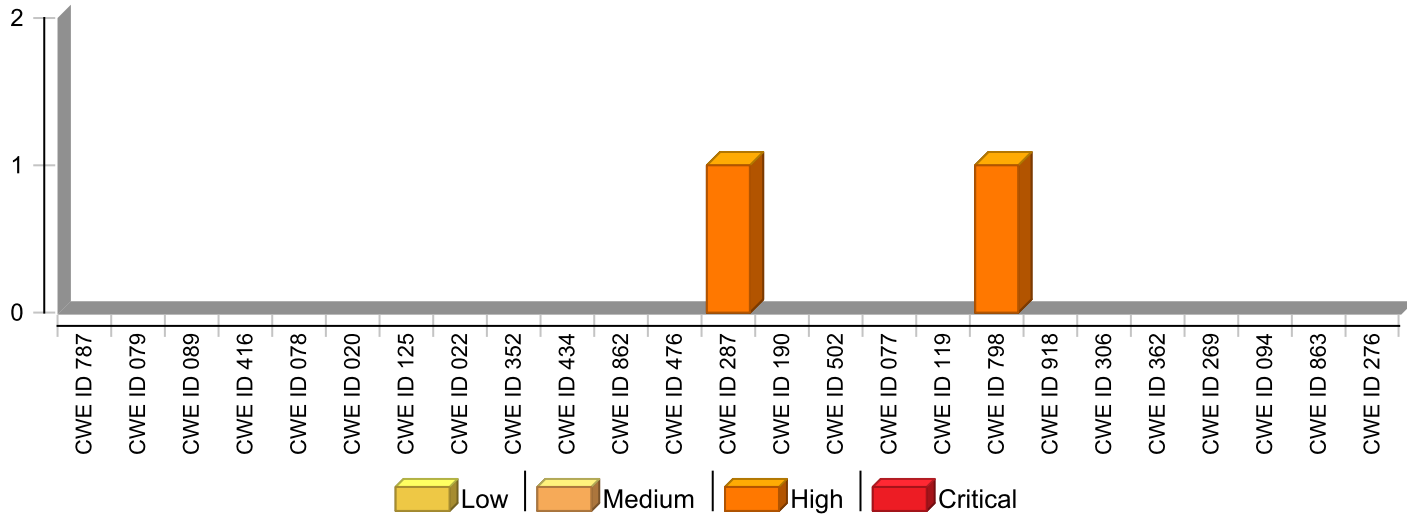
The 2023 CWE Top 25 Most Dangerous Software Errors lists the most widespread and critical weaknesses that can lead to serious vulnerabilities in software (as demonstrated by the National Vulnerability Database). These weaknesses occur frequently, are often easy to find, and easy to exploit. They are dangerous because they will frequently enable attackers to completely take over the software, steal data, or prevent the software from working at all. The list is the result of heuristic formula that the CWE Team used with a data-driven approach that leveraged the Common Vulnerabilities and Exposure (CVE), National Vulnerability Database (NVD), and Common Vulnerability Scoring System (CVSS). Due to the hierarchical nature of the CWE taxonomy, Fortify considers all CWE IDs which are children of a Top 25 entry, as included within the context of the entry due to the "CHILD-OF" relationship within the hierarchy. Exercise caution if using only this Top 25 list to prioritize auditing efforts because the software under analysis might not align with the assumptions of the heuristic used to define the Top 25. For example, many of these weaknesses are related to C-like languages and the software under analysis might not be within the C-family of languages - thus, many CWEs would not be in scope.

Project Name:	DADAGAS
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present
Remediation Effort (Hrs):	0.2

Issues by Priority



Issues by CWE Top 25 2023 Categories



* The detailed sections following the Executive Summary contain specifics.

Project Description

This section provides an overview of the Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	Feb 22, 2026 9:49 PM	**Engine Version:**	23.2.0.0125
Host Name:	DESKTOP-0H4L7MQ	**Certification:**	VALID
Number of Files:	1	**Lines of Code:**	95

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Community, Cloud	2025.3.0.0007
Fortify Secure Coding Rules, Community, Universal	2025.3.0.0007
Fortify Secure Coding Rules, Core, Cloud	2025.3.0.0007
Fortify Secure Coding Rules, Core, Python	2025.3.0.0007
Fortify Secure Coding Rules, Core, Universal	2025.3.0.0007
Fortify Secure Coding Rules, Extended, Configuration	2025.3.0.0007
Fortify Secure Coding Rules, Extended, Content	2025.3.0.0007
Fortify Secure Coding Rules, Community, Cloud	2024.2.0.0008
Fortify Secure Coding Rules, Community, Universal	2024.2.0.0008
Fortify Secure Coding Rules, Core, Cloud	2024.2.0.0008
Fortify Secure Coding Rules, Core, Python	2024.2.0.0008
Fortify Secure Coding Rules, Core, Universal	2024.2.0.0008
Fortify Secure Coding Rules, Extended, Configuration	2024.2.0.0008
Fortify Secure Coding Rules, Extended, Content	2024.2.0.0008

Issue Breakdown

The following table summarizes the number of issues identified across the different CWE Top 25 2023 categories and broken down by Fortify Priority Order.

	Fortify Priority				Total Issues	Effort (hrs)
	Critical	High	Medium	Low		
[1] CWE ID 787	0	0	0	0	0	0.0
[2] CWE ID 079	0	0	0	0	0	0.0
[3] CWE ID 089	0	0	0	0	0	0.0
[4] CWE ID 416	0	0	0	0	0	0.0
[5] CWE ID 078	0	0	0	0	0	0.0
[6] CWE ID 020	0	0	0	0	0	0.0
[7] CWE ID 125	0	0	0	0	0	0.0
[8] CWE ID 022	0	0	0	0	0	0.0
[9] CWE ID 352	0	0	0	0	0	0.0
[10] CWE ID 434	0	0	0	0	0	0.0
[11] CWE ID 862	0	0	0	0	0	0.0
[12] CWE ID 476	0	0	0	0	0	0.0
[13] CWE ID 287	0	1	0	0	1	0.2
[14] CWE ID 190	0	0	0	0	0	0.0
[15] CWE ID 502	0	0	0	0	0	0.0
[16] CWE ID 077	0	0	0	0	0	0.0
[17] CWE ID 119	0	0	0	0	0	0.0
[18] CWE ID 798	0	1	0	0	1	0.2
[19] CWE ID 918	0	0	0	0	0	0.0
[20] CWE ID 306	0	0	0	0	0	0.0
[21] CWE ID 362	0	0	0	0	0	0.0
[22] CWE ID 269	0	0	0	0	0	0.0
[23] CWE ID 094	0	0	0	0	0	0.0
[24] CWE ID 863	0	0	0	0	0	0.0
[25] CWE ID 276	0	0	0	0	0	0.0

NOTE:
1. Reported issues in the above table may violate more than one CWE Top 25 2023 category. As such, the same issue may appear in more than one row. The total number of unique vulnerabilities are reported in the Executive Summary table.
2. For the same reason, the Project-level remediation effort total shown in the Executive Summary removes the effect of any duplication and may be smaller than the sum of the remediation effort per individual category.
3. Similarly, the remediation effort per external category is not intended to equal the sum of the remediation effort from the issue details section since individual files may contain issues in multiple Fortify priorities or audit folders.

Issue Details

Below is an enumeration of all issues found in the project. The issues are organized by CWE Top 25 2023, Fortify Priority Order, and vulnerability category. The issues are then further broken down by the package, namespace, or location in which they occur. Issues reported at the same line number with the same category originate from different taint sources.

[1] CWE ID 787

CWE-787 is used to identify an "Out-of-bounds Write" weakness.

These weaknesses occur because "The software writes data past the end, or before the beginning, of the intended buffer."

No Issues

[2] CWE ID 079

CWE-79 is used to identify an "Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')" weakness.

These weaknesses occur because "The 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."

No Issues

[3] CWE ID 089

CWE-89 is used to identify an "Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')" weakness.

These weaknesses occur because "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."

No Issues

[4] CWE ID 416

CWE-416 is used to identify a "Use After Free" weakness.

These weaknesses occur because "Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code."

No Issues

[5] CWE ID 078

CWE-78 is used to identify an "Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')" weakness.

These weaknesses occur because "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."

No Issues

[6] CWE ID 020

CWE-20 is used to identify an "Improper Input Validation" weakness.

These weaknesses occur because "The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program."

No Issues

[7] CWE ID 125

CWE-125 is used to identify an "Out-of-bounds Read" weakness.

These weaknesses occur because "The software reads data past the end, or before the beginning, of the intended buffer."

No Issues

[8] CWE ID 022

CWE-22 is used to identify an "Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')" weakness.

These weaknesses occur because "The 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."

No Issues

[9] CWE ID 352

CWE-352 is used to identify a "Cross-Site Request Forgery (CSRF)" weakness.

These weaknesses occur because "The 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."

No Issues

[10] CWE ID 434

CWE-434 is used to identify an "Unrestricted Upload of File with Dangerous Type" weakness.

These weaknesses occur because "The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment."

No Issues

[11] CWE ID 862

CWE-862 is used to identify a "Missing Authorization" weakness.

These weaknesses occur because "The software does not perform an authorization check when an actor attempts to access a resource or perform an action."

No Issues

[12] CWE ID 476

CWE-476 is used to identify a "NULL Pointer Dereference" weakness.

"A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit."

No Issues

[13] CWE ID 287

CWE-287 is used to identify an "Improper Authentication" weakness.

These weaknesses occur because "When an actor claims to have a given identity, the software does not prove or insufficiently proves that the claim is correct."

Key Management: Hardcoded Encryption Key ***Remediation Effort(Hrs): 0.2***		**High**
Package: <none>		
Location	**Analysis Info**	**Analyzer**
SCATest.py:13	**Sink:** FieldAccess: secret_key **Enclosing Method:** () **Source:**	SCA

[14] CWE ID 190

CWE-190 is used to identify an "Integer Overflow or Wraparound" weakness.

These weaknesses occur because "The 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."

No Issues

[15] CWE ID 502

CWE-502 is used to identify a "Deserialization of Untrusted Data" weakness.

These weaknesses occur because "The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid."

No Issues

[16] CWE ID 077

CWE-77 is used to identify an "Improper Neutralization of Special Elements used in a Command ('Command Injection')" weakness.

These weaknesses occur because "The software constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component."

No Issues

[17] CWE ID 119

CWE-119 is used to identify an "Improper Restriction of Operations within the Bounds of a Memory Buffer" weakness.

These weaknesses occur because "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."

No Issues

[18] CWE ID 798

CWE-798 is used to identify a "Use of Hard-coded Credentials)" weakness.

These weaknesses occur because "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."

Key Management: Hardcoded Encryption Key ***Remediation Effort(Hrs): 0.2***		**High**
Package: <none>		
Location	**Analysis Info**	**Analyzer**
SCATest.py:13	**Sink:** FieldAccess: secret_key **Enclosing Method:** () **Source:**	SCA

[19] CWE ID 918

CWE-918 is used to identify a "Server-Side Request Forgery (SSRF)" weakness.

These weaknesses occur because "The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination."

No Issues

[20] CWE ID 306

CWE-306 is used to identify a "Missing Authentication for Critical Function" weakness.

These weaknesses occur because "The software does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources."

No Issues

[21] CWE ID 362

CWE-362 is used to identify a "Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')" weakness.

These weaknesses occur because "The program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently."

No Issues

[22] CWE ID 269

CWE-269 is used to identify an "Improper Privilege Management" weakness.

These weaknesses occur because "The software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor."

No Issues

[23] CWE ID 094

CWE-94 is used to identify an "Improper Control of Generation of Code ('Code Injection')" weakness.

These weaknesses occur because "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."

No Issues

[24] CWE ID 863

CWE-863 is used to identify an "Incorrect Authorization" weakness.

The software performs an authorization check when an actor attempts to access a resource or perform an action, but it does not correctly perform the check. This allows attackers to bypass intended access restrictions".

No Issues

[25] CWE ID 276

CWE-276 is used to identify an "Incorrect Default Permissions" weakness.

These weaknesses occur because "The product, upon installation, sets incorrect permissions for an object that exposes it to an unintended actor."

No Issues

Description of Key Terminology

Likelihood and Impact

Likelihood
Likelihood is the probability that a vulnerability will be accurately identified and successfully exploited.

Impact
Impact is the potential damage an attacker could do to assets by successfully exploiting a vulnerability. This damage can be in the form of, but not limited to, financial loss, compliance violation, loss of brand reputation, and negative publicity.

Fortify Priority Order

Critical
Critical-priority issues have high impact and high likelihood. Critical-priority issues are easy to detect and exploit and result in large asset damage. These issues represent the highest security risk to the application. As such, they should be remediated immediately.

SQL Injection is an example of a critical issue.

High
High-priority issues have high impact and low likelihood. High-priority issues are often difficult to detect and exploit, but can result in large asset damage. These issues represent a high security risk to the application. High-priority issues should be remediated in the next scheduled patch release.

Password Management: Hardcoded Password is an example of a high issue.

Medium
Medium-priority issues have low impact and high likelihood. Medium-priority issues are easy to detect and exploit, but typically result in small asset damage. These issues represent a moderate security risk to the application. Medium-priority issues should be remediated in the next scheduled product update.

Path Manipulation is an example of a medium issue.

Low
Low-priority issues have low impact and low likelihood. Low-priority issues can be difficult to detect and exploit and typically result in small asset damage. These issues represent a minor security risk to the application. Low-priority issues should be remediated as time allows.

Dead Code is an example of a low issue.

Remediation Effort

The report provides remediation effort estimates. You can use these estimates to perform a relative comparison of projects and as a starting point for estimates specific to your organization. Remediation effort estimates are provided in the following report sections:

- Executive Summary
- Issue Breakdown
- Issue Details

To determine remediation effort for a collection of issues, Software Security Center weights each issue based on its category (“remediation constant”) and adds an overhead calculation based on the number of distinct files which contain the set of issues. The formula used at each report level is the same:

- Remediation Effort (in mins) = SUM(remediation constant for each issue in the set) + 6 * Number of distinct files in that set of issues.

At the lowest level of detail, issues are grouped based on Fortify category and Fortify priority OR Fortify category and folder name, depending on report options. So, for example, the Issue Details section of the report might show the remediation effort for “SQL Injection, Critical” or “SQL Injection, MyFolder”.

At the Issue Breakdown level, remediation effort is shown at the level of each external (non-Fortify) category (such as “AC-3 Access Enforcement” in the case of NIST, or “A1 Unvalidated Input” in the case of OWASP Top10). Remediation effort is calculated for the set of all issues that fall into that external category (irrespective of Fortify priority or folder name). As an example, if there are two SQL injection vulnerabilities, one critical and one medium, within the same file, the file overhead is only included once.

At the Executive Summary level, all issues of that project which are mapped to the specified external category list (such as NIST or CWE) are used in the remediation effort calculation.

Fortify recommends that you treat the different levels of remediation effort as information relevant at that level only. You cannot add up remediation effort at a lower level and expect it to match the remediation effort at a higher level.

About Fortify Solutions

Fortify is the leader in end-to-end application security solutions with the flexibility of testing on-premise and on-demand to cover the entire software development lifecycle. Learn more at www.microfocus.com/solutions/application-security.