# Coverity® **Security Report**

## ALPR_ORIG
## v. original

| | |
|---|---|
| Enterprise | LG Electronics |
| Division | Development Team |
| Assurance Level | AL1 (90) |
| Severity Mapping | Carrier Grade |
| Prepared For | cmu studio project |
| Prepared By | Team2 AhnLab |
| Prepared On | Jul 3, 2022 10:23 AM |

## Executive Summary

This report details the application security assessment activities that were carried out, providing a summary of findings, compliance against published policy requirements, and remediation actions required. Also provided is a detailed breakdown and cross reference between technical findings and Coverity analysis results.

The intended audience for this report is an application security assurance team and their clients or end users. To review detailed code-level findings, it is recommended that developers click this link to the Coverity Connect platform (http://cim.lge.com:5500/reports#p10080) in order to see source code annotated with remediation recommendations.

Lines of Code Inspected: 707879

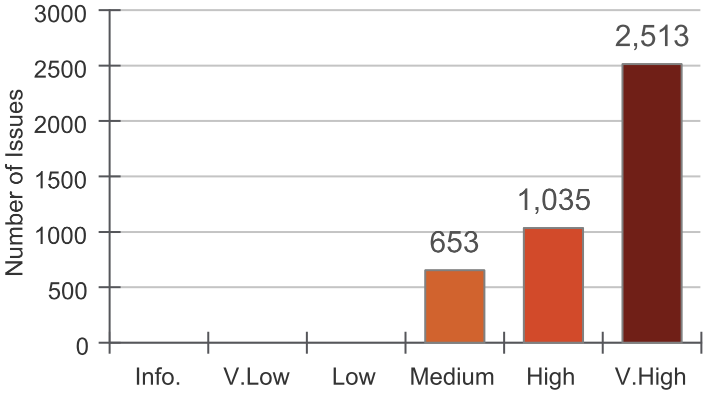
### Scorecard

The issues were evaluated according to each element of the report's policy. The results are shown in the table below. An overall status of "pass" is assigned if all the policy elements passed.

| Policy Element | Target | Value | Passed |
|---|---|---|---|
| Security Score | 90 | 38 | No |
| OWASP Top 10 Count | 0 | 0 | Yes |
| CWE/SANS Top 25 Count | 0 | 2563 | No |
| Analysis Date | 2022-6-3 | 2022-7-2 | Yes |
| **Overall Status** | | | **No** |

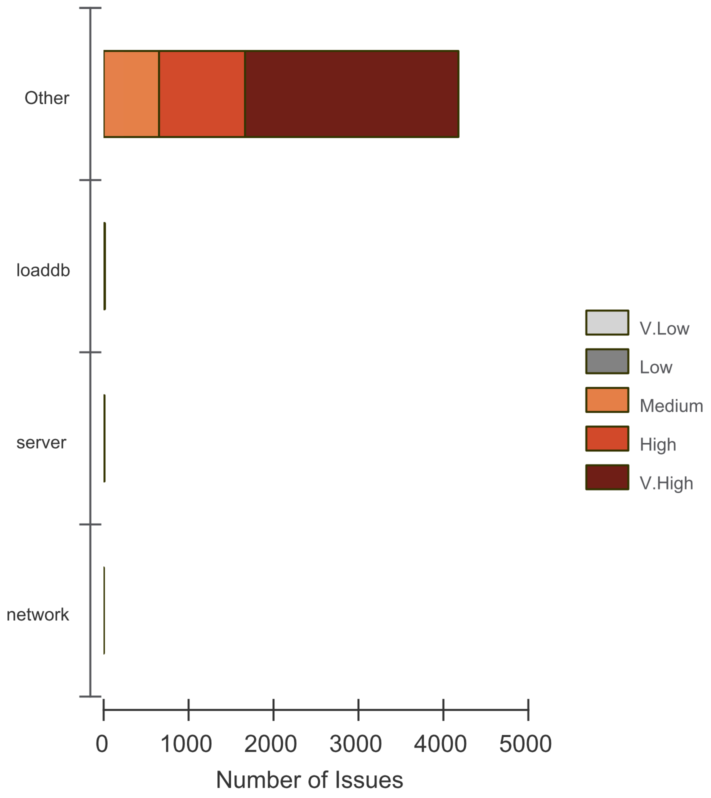
### Issues By Severity

A total of 4201 security issues were found. Each issue was given a severity based on the severity mapping. The chart below shows the number of occurrences of each of the six severity values.



### Severity By Component

Issues are shown grouped by severity and counted by Component.



### Additional Quality Measures

This table reports the numbers of issues of various categories that were not included in the Security Score calculation. Although they were excluded from the report, they may nonetheless indicate the presence of significant quality or security issues. Issues which do not have CWE number or Technical impact are counted as Non-Security issues.

| Category | Count |
|---|---|
| Issues Marked "False Positive" or "Intentional" | 0 |
| Non-Security Issues | 1710 |
| Issues Scored as "Informational" | 0 |

## Action Items

The code base was evaluated based on the policy in force. The policy has the following elements:

- The Security Score must meet or exceed the target set by the Assurance Level. See the Security Details section for more information.
- There must be no OWASP Top 10 issues among those found in the project. See the OWASP Top 10 section for details.
- There must be no CWE/SANS Top 25 issues among those found in the project. See the CWE/SANS Top 25 section for details.
- All snapshots must have been analyzed within 30 days. See the Analysis Details section for more information.

Coverity recommends the following actions in order to resolve critical outstanding issues, achieve compliance with policy, and improve the overall security of the software.

### Security Score Remediation

Resolve issues that contribute to a substandard security score. Resolving the issues below will improve the security score from 38 to 90:

- 2513 “Very high” issues.
- 1035 “High” issues.
- 651 “Medium” issues.

### OWASP Top 10 Remediation

The project has no issues in the OWASP Top 10.

### CWE/SANS Top 25 Remediation

Resolve 2563 issues that are present in the CWE/SANS Top 25. See the CWE/SANS Top 25 Section for a list of them.

### Recent Source Code Analysis

Regular source code analysis is key to identifying security issues in a timely manner and to ensuring that these issues are effectively eliminated, in-line with development activities.

The current results are sufficiently recent (less than 30 days old).

### Long Term and Residual Risk Management

Review and consider broader improvement to the overall security posture of the target application.
Review outstanding lesser-rated issues to ensure minimal residual risk.
Review issues marked false positive to be sure that a coding change will not eliminate them.
Review any security issues marked Informational to see if some are in fact credible threats.
Review and correct non-security issues found by Coverity Analysis, in order to increase the overall quality of the code.

## Security Details

The severity mapping shows how technical impacts (possible security flaws) are paired with severities.  This severity mapping table also shows the number of issues for each technical impact.

| | |
|---|---|
| **Severity Mapping Name:** | Carrier Grade |
| **Severity Mapping Description:** | Very stringent |

| Technical Impact | Severity | Number of Issues |
|---|---|---|
| Execute unauthorized code | Very high | 2,491 |
| Gain privileges | Very high | 22 |
| Bypass protection mechanism | High | 0 |
| Denial of service, unreliable execution | High | 1,015 |
| Modify data | High | 20 |
| Denial of service, Resource consumption | Medium | 623 |
| Hide activities | Medium | 0 |
| Read data | Medium | 30 |
| **Total** | | **4201** |

## Analysis Details

A Coverity project is a collection of one or more streams containing separately-analyzed snapshots.  The latest snapshot in each stream is used when reporting results for a project. This section gives details about the streams and the analysis performed for each snapshot.

| Stream Name | Snapshot ID | Analysis Date | Analysis Version | Target |
|---|---|---|---|---|
| ALPR_ORIG_STATIC | 10455 | 2022-7-2 11:26:27 오후 | 2022.3.3 | |

## The 2017 OWASP Top 10 List

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. The OWASP maintains the OWASP Top 10 List for 2017, a prioritized list of security weaknesses. OWASP says, “We can no longer afford to tolerate relatively simple security problems like those presented in this OWASP Top 10.”

Each entry in the OWASP Top 10 refers to a set of CWE entries. Those entries may be individual weaknesses or families of weaknesses. See the next section for further discussion.

The table below shows the number of issues found in each category of the OWASP Top 10 for 2017.

| 2017 OWASP Top 10 Categories | CWE Number | Count |
|---|---|---|
| **1**. Injection | 1027 | 0 |
| **2**. Broken Authentication | 1028 | 0 |
| **3**. Sensitive Data Exposure | 1029 | 0 |
| **4**. XML External Entities (XXE) | 1030 | 0 |
| **5**. Broken Access Control | 1031 | 0 |
| **6**. Security Misconfiguration | 1032 | 0 |
| **7**. Cross-Site Scripting (XSS) | 1033 | 0 |
| **8**. Insecure Deserialization | 1034 | 0 |
| **9**. Using Components with Known Vulnerabilities * | 1035 | 0 |
| **10**. Insufficient Logging & Monitoring | 1036 | 0 |
| **Total** | | **0** |

* Category 9 of the OWASP Top 10 for 2017, "Using Components with Known Vulnerabilities," is not detected by Coverity Static Analysis, but is detected by BlackDuck and Protecode ES, which are other Synopsys products.

## The 2019 CWE/SANS Top 25 List

The Common Weakness Enumeration is a community-developed dictionary of software weakness types. The 2019 CWE/SANS Top 25 Most Dangerous Software Errors (or, "Top 25") is a list of weaknesses, taken from the CWE, that are thought to be the most widespread and critical errors that can lead to serious vulnerabilities in software.

Each category in the Top 25 List mentions one primary CWE identifier (CWE ID). Such a CWE ID can refer to an individual weakness or to a family of related weaknesses, since a given CWE ID may have children CWE IDs, which in turn may have children CWE IDs of their own. A Coverity issue corresponds to the most relevant CWE ID. A CWE/SANS Top 25 Category will consist of all of the Coverity issues that correspond to either the mentioned CWE ID or to one of its associated descendants.

The table below lists all the entries of the Top 25 and shows how many Coverity issues in the current project were found to be members of the Top 25.

| 2019 CWE/SANS Top 25 Categories | CWE Number | Count |
|---|---|---|
| **1**. Improper Restriction of Operations within the Bounds of a Memory Buffer | CWE-119 | 1538 |
| **2**. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | CWE-79 | 0 |
| **3**. Improper Input Validation | CWE-20 | 49 |
| **4**. Information Exposure | CWE-200 | 0 |
| **5**. Out-of-bounds Read | CWE-125 | 30 |
| **6**. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') | CWE-89 | 0 |
| **7**. Use After Free | CWE-416 | 493 |
| **8**. Integer Overflow or Wraparound | CWE-190 | 80 |
| **9**. Cross-Site Request Forgery (CSRF) | CWE-352 | 0 |
| **10**. Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') | CWE-22 | 0 |
| **11**. Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') | CWE-78 | 0 |
| **12**. Out-of-bounds Write | CWE-787 | 0 |
| **13**. Improper Authentication | CWE-287 | 0 |
| **14**. NULL Pointer Dereference | CWE-476 | 326 |
| **15**. Incorrect Permission Assignment for Critical Resource | CWE-732 | 0 |
| **16**. Unrestricted Upload of File with Dangerous Type | CWE-434 | 0 |
| **17**. Improper Restriction of XML External Entity Reference ('XXE') | CWE-611 | 0 |
| **18**. Improper Control of Generation of Code ('Code Injection') | CWE-94 | 0 |
| **19**. Use of Hard-coded Credentials | CWE-798 | 0 |
| **20**. Uncontrolled Resource Consumption ('Resource Exhaustion') | CWE-400 | 42 |
| **21**. Missing Release of Resource after Effective Lifetime | CWE-772 | 5 |
| **22**. Untrusted Search Path | CWE-426 | 0 |
| **23**. Deserialization of Untrusted Data | CWE-502 | 0 |
| **24**. Improper Privilege Management | CWE-269 | 0 |
| **25**. Improper Certificate Validation | CWE-295 | 0 |
| **Total** | | **2563** |

## Detailed Issues Ranked By Severity

**Showing 200 of 4,201 issues with valid CWE entries.**

## Severity: Very high

### Technical Impact: Execute unauthorized code

**CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer**
This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681654<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/scalelow.c:1992 | Other |
| 681650<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jfdctint.c:3552 | Other |
| 681649<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/zlib/src/inftrees.c:256 | Other |
| 681642<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/ratngs.h:467 | Other |
| 305544<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/tclap/ValueArg.h:336 | Other |
| 681635<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/statistc.cpp:682 | Other |
| 681621<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/intproto.cpp:512 | Other |
| 681622<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libpng/src/pngpread.c:1520 | Other |
| 681616<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/grayquant.c:1440 | Other |
| 681611<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_predict.c:547 | Other |
| 681610<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/trainingsample.h:159 | Other |
| 681606<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/opencv/build/include/opencv2/core/matx.hpp:1130 | Other |
| 681607<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/fmorphgenlow.1.c:5767 | Other |
| 681605<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptive.cpp:64 | Other |
| 681596<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/dgif_lib.c:820 | Other |
| 681579<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/detlinefit.cpp:82 | Other |

## Technical Impact: Execute unauthorized code

### CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681569<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/intproto.cpp:314 | Other |
| 681562<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pix5.c:2601 | Other |
| 681559<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorcontent.c:1014 | Other |
| 681553<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/getarg.c:464 | Other |
| 681550<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jcparam.c:44 | Other |
| 681544<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/grayquantlow.c:849 | Other |
| 681541<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jfdctint.c:1440 | Other |
| 681526<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/cube/bmp_8.cpp:559 | Other |
| 681504<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/morphapp.c:1403 | Other |
| 681501<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/wordrec/pieces.cpp:300 | Other |
| 681495<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_ojpeg.c:2161 | Other |
| 681485<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/intfeaturedist.cpp:88 | Other |
| 681479<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/colfind.cpp:851 | Other |
| 681472<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/cube/cube_search_object.cpp:441 | Other |
| 681469<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jquant2.c:376 | Other |
| 681467<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jcmarker.c:245 | Other |
| 681464<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/zlib/src/trees.c:539 | Other |
| 681457<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/support/re2/util/sparse_set.h:146 | Other |
| 681456<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jquant1.c:672 | Other |

## Technical Impact: Execute unauthorized code

### CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681453<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/osdetect.cpp:496 | Other |
| 681445<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/featdefs.cpp:279 | Other |
| 681444<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/intproto.cpp:512 | Other |
| 681442<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/unicharset.h:395 | Other |
| 681435<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorquant1.c:1791 | Other |
| 681429<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/viewer/scrollview.cpp:735 | Other |
| 681421<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/dict/dawg.h:500 | Other |
| 681414<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/zlib/src/deflate.c:1914 | Other |
| 681413<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/getarg.c:476 | Other |
| 681406<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdhuff.c:1236 | Other |
| 681389<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/matrix.h:80 | Other |
| 681383<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/numafunc1.c:1699 | Other |
| 681371<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/utils.c:665 | Other |
| 681363<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/intmatcher.cpp:856 | Other |
| 681364<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/recogdid.c:495 | Other |
| 681359<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libpng/src/pngrtran.c:994 | Other |
| 681360<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/makerow.cpp:945 | Other |
| 681358<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/statistc.cpp:548 | Other |
| 681353<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/ocr/segmentation/histogram.cpp:141 | Other |

## Technical Impact: Execute unauthorized code

### CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681347<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/applybox.cpp:537 | Other |
| 681336<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/zlib/src/trees.c:1028 | Other |
| 681109<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/oldbasel.cpp:629 | Other |
| 681106<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/ratngs.h:314 | Other |
| 681104<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/genericvector.h:1024 | Other |
| 681101<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/fmorphgenlow.1.c:5493 | Other |
| 681100<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/opencv/build/include/opencv2/flann/lsh_table.h:378 | Other |
| 681099<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/cube/cube_line_segmenter.cpp:156 | Other |
| 681089<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libpng/src/pngrtran.c:993 | Other |
| 681090<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/unicharset.h:415 | Other |
| 681086<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/utils.c:2799 | Other |
| 681083<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/statistc.cpp:411 | Other |
| 681080<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/oldbasel.cpp:749 | Other |
| 681079<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jfdctint.c:3881 | Other |
| 681075<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/imagefind.cpp:971 | Other |
| 681326<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/pithsync.cpp:569 | Other |
| 681319<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/genericvector.h:243 | Other |
| 681317<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdhuff.c:420 | Other |
| 681313<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/mod128.cpp:98 | Other |

## Technical Impact: Execute unauthorized code

### CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681296<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorquant1.c:3593 | Other |
| 681292<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/convolvelow.c:130 | Other |
| 681293<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorcontent.c:256 | Other |
| 681286<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdhuff.c:817 | Other |
| 681284<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/quantize.c:310 | Other |
| 681282<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdhuff.c:1111 | Other |
| 681280<br>Out-of-bounds write | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/dgif_lib.c:820 | Other |
| 681273<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/unicharset.h:410 | Other |
| 681268<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/support/re2/util/sparse_array.h:418 | Other |
| 681256<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/coloring.c:779 | Other |
| 681252<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/maze.c:409 | Other |
| 681248<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libpng/src/pngrtran.c:392 | Other |
| 681247<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdarith.c:709 | Other |
| 681245<br>Out-of-bounds access | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_dirwrite.c:272 | Other |
| 681246<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/dict/trie.cpp:697 | Other |
| 681239<br>CERT-CPP Containers | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/strngs.cpp:424 | Other |

## Technical Impact: Execute unauthorized code

### CWE 120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.

**Details:** A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the "classic" case in which the program copies the buffer without restricting how much is copied. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections.more details.

**Remediation:** Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681648<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jchuff.c:427 | Other |
| 681597<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/getarg.c:481 | Other |
| 681497<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdarith.c:269 | Other |
| 681454<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/cube/search_node.h:131 | Other |
| 681447<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/quantize.c:312 | Other |
| 681437<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/getarg.c:476 | Other |
| 681425<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/imagefind.cpp:972 | Other |
| 681379<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jcdctmgr.c:73 | Other |
| 681097<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/giflib/src/getarg.c:458 | Other |
| 681095<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rank.c:420 | Other |
| 681333<br>Copy into fixed size buffer | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/dict/dict.cpp:615 | Other |
| 681309<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/cluster.cpp:1194 | Other |
| 681304<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptmatch.cpp:746 | Other |
| 681301<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jmemmgr.c:982 | Other |
| 681300<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptmatch.cpp:1822 | Other |

## Technical Impact: Execute unauthorized code

### CWE 129: Improper Validation of Array Index

This CWE entry is at position 3 in the CWE/SANS Top 25.

**Summary:** The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.

**Remediation:** Use an input validation framework such as Struts or the OWASP ESAPI Validation API. If you use Struts, be mindful of weaknesses covered by the CWE-101 category.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681631<br>CERT-CPP Characters and Strings | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/support/utf8.cpp:18 | Other |

### CWE 190: Integer Overflow or Wraparound

This CWE entry is at position 8 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected. This is especially the case if the integer overflow can be triggered using user-supplied inputs. This becomes security-critical when the result is used to control looping, make a security decision, or determine the offset or size in behaviors such as memory allocation, copying, concatenation, etc.more details.

**Remediation:** Ensure that all protocols are strictly defined, such that all out-of-bounds behavior can be identified simply, and require strict conformance to the protocol.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681573<br>Integer overflow warning | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/dict/dawg.cpp:212 | Other |
| 681572<br>Integer overflowed argument | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/sarray.c:1525 | Other |
| 681460<br>Overflowed constant | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdarith.c:73 | Other |
| 681430<br>Overflowed constant | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/jpegio.c:1197 | Other |
| 681108<br>Integer overflow warning | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/wordrec/language_model.cpp:1031 | Other |
| 681087<br>Integer overflow warning | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/dict/trie.h:251 | Other |

## Technical Impact: Execute unauthorized code

### CWE 20: Improper Input Validation

This CWE entry is at position 3 in the CWE/SANS Top 25.

**Summary:** The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program.

**Details:** When software does not validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.more details.

**Remediation:** Use an input validation framework such as Struts or the OWASP ESAPI Validation API. If you use Struts, be mindful of weaknesses covered by the CWE-101 category.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681514<br>Untrusted value as argument | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/tessdatamanager.cpp:57 | Other |
| 681373<br>Untrusted value as argument | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptive.cpp:438 | Other |

### CWE 394: Unexpected Status Code or Return Value

**Summary:** The software does not properly check when a function or operation returns a value that is legitimate for the function, but is not expected by the software.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681598<br>Improper use of negative value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/sampleiterator.cpp:96 | Other |

### CWE 415: Double Free

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.

**Details:** When a program calls free() twice with the same argument, the program's memory management data structures become corrupted. This corruption can cause the program to crash or, in some circumstances, cause two later calls to malloc() to return the same pointer. If malloc() returns the same value twice and the program later gives the attacker control over the data that is written into this doubly-allocated memory, the program becomes vulnerable to a buffer overflow attack.more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681660<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/convolve.c:1159 | Other |
| 681619<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/ccbord.c:763 | Other |
| 681470<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/morphapp.c:837 | Other |
| 681330<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:191 | Other |

## Technical Impact: Execute unauthorized code

### CWE 415: Double Free

This CWE entry is at position 1 in the CWE/SANS Top 25.

**Summary:** The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.

**Details:** When a program calls free() twice with the same argument, the program's memory management data structures become corrupted. This corruption can cause the program to crash or, in some circumstances, cause two later calls to malloc() to return the same pointer. If malloc() returns the same value twice and the program later gives the attacker control over the data that is written into this doubly-allocated memory, the program becomes vulnerable to a buffer overflow attack.more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681316<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/convertfiles.c:122 | Other |
| 681303<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/grayquant.c:415 | Other |
| 681285<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/psio1.c:643 | Other |
| 681264<br>Double free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pix4.c:3046 | Other |

### CWE 416: Use After Free

This CWE entry is at position 7 in the CWE/SANS Top 25.

**Summary:** Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

**Details:** 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. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681652<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/viewfiles.c:180 | Other |
| 681653<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/enhance.c:1078 | Other |
| 681643<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/convolve.c:155 | Other |
| 681638<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/morphseq.c:682 | Other |
| 681617<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/recogtrain.c:250 | Other |
| 681609<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/seedfill.c:251 | Other |

## Technical Impact: Execute unauthorized code

### CWE 416: Use After Free

This CWE entry is at position 7 in the CWE/SANS Top 25.

**Summary:** Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

**Details:** 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. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681603<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorcontent.c:916 | Other |
| 681600<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/affine.c:835 | Other |
| 681601<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/scale.c:344 | Other |
| 681599<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/affine.c:1558 | Other |
| 681571<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/baseline.c:244 | Other |
| 681532<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/dewarp3.c:180 | Other |
| 681512<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/seedfill.c:458 | Other |
| 681499<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/wordseg.cpp:84 | Other |
| 681498<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:585 | Other |
| 681494<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/ccbord.c:834 | Other |
| 681481<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pixafunc1.c:1679 | Other |
| 681475<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/adaptmap.c:1010 | Other |
| 681466<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pageseg.c:419 | Other |
| 681433<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/compare.c:1538 | Other |
| 681418<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/morphapp.c:837 | Other |
| 681408<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/enhance.c:913 | Other |
| 681400<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/runlength.c:150 | Other |
| 681386<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pix5.c:2307 | Other |
| 681384<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:584 | Other |

## Technical Impact: Execute unauthorized code

### CWE 416: Use After Free

This CWE entry is at position 7 in the CWE/SANS Top 25.

**Summary:** Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

**Details:** 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. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681365<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:582 | Other |
| 681356<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/blend.c:1960 | Other |
| 681354<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorseg.c:169 | Other |
| 681349<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/ptafunc1.c:2077 | Other |
| 681112<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/enhance.c:918 | Other |
| 681110<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/readbarcode.c:234 | Other |
| 681107<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/ccthin.c:298 | Other |
| 681096<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/psio2.c:1472 | Other |
| 681093<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/enhance.c:913 | Other |
| 681081<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/warper.c:683 | Other |
| 681335<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/baseline.c:244 | Other |
| 681331<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/convertfiles.c:132 | Other |
| 681324<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/blend.c:1883 | Other |
| 681308<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pixcomp.c:1655 | Other |
| 681298<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/seedfill.c:373 | Other |
| 681297<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/colfind.cpp:962 | Other |
| 681291<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/tesseractclass.cpp:760 | Other |
| 681289<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pix4.c:2960 | Other |
| 681288<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/scale.c:1661 | Other |

## Technical Impact: Execute unauthorized code

### CWE 416: Use After Free

This CWE entry is at position 7 in the CWE/SANS Top 25.

**Summary:** Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

**Details:** 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. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:more details.

**Remediation:** Choose a language that provides automatic memory management.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681283<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:192 | Other |
| 681278<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/pix4.c:2205 | Other |
| 681265<br>CERT-CPP Memory Management | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/dewarp3.c:179 | Other |
| 681259<br>Use after free | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/rotate.c:559 | Other |
| 681242<br>CERT-CPP Expressions | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/affine.c:1344 | Other |

### CWE 476: NULL Pointer Dereference

This CWE entry is at position 14 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** NULL pointer dereference issues can occur through a number of flaws, including race conditions, and simple programming omissions.more details.

**Remediation:** If all pointers that could have been modified are sanity-checked previous to use, nearly all NULL pointer dereferences can be prevented.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681614<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/docqual.cpp:186 | Other |
| 681595<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/cjson.c:555 | Other |
| 681589<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/support/re2/prefilter_tree.cc:282 | Other |
| 681538<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptmatch.cpp:463 | Other |
| 681520<br>Dereference after null check | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/equationdetect.cpp:230 | Other |
| 681516<br>Dereference after null check | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/blobbox.cpp:101 | Other |

## Technical Impact: Execute unauthorized code

### CWE 476: NULL Pointer Dereference

This CWE entry is at position 14 in the CWE/SANS Top 25.

**Summary:** 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.

**Details:** NULL pointer dereference issues can occur through a number of flaws, including race conditions, and simple programming omissions.more details.

**Remediation:** If all pointers that could have been modified are sanity-checked previous to use, nearly all NULL pointer dereferences can be prevented.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681517<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/support/re2/prefilter.cc:335 | Other |
| 681506<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorspace.c:895 | Other |
| 681486<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/seedfilllow.c:1102 | Other |
| 681484<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/alpr_impl.cpp:529 | Other |
| 681483<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/paragraphs.cpp:115 | Other |
| 681432<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/osdetect.cpp:196 | Other |
| 681424<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libpng/src/pngpread.c:1419 | Other |
| 681396<br>Dereference after null check | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libjpeg/src/jdatadst.c:136 | Other |
| 681394<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_dirread.c:1254 | Other |
| 681392<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccmain/fixspace.cpp:350 | Other |
| 681391<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/bmpio.c:636 | Other |
| 681387<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccutil/unicharset.cpp:992 | Other |
| 681370<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/cluster.cpp:2629 | Other |
| 681355<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/colorquant2.c:629 | Other |
| 681113<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/fpix2.c:1003 | Other |
| 681102<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/textord/tablefind.cpp:465 | Other |
| 681076<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/ccstruct/coutln.cpp:202 | Other |
| 681261<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/tesseract/classify/adaptmatch.cpp:576 | Other |
| 681250<br>Dereference null return value | /OpenAPLR-3.1.1_Vs2022/libs/libopenalpr/cjson.c:554 | Other |

## Technical Impact: Execute unauthorized code

### CWE 484: Omitted Break Statement in Switch

**Summary:** The program omits a break statement within a switch or similar construct, causing code associated with multiple conditions to execute. This can cause problems when the programmer only intended to execute code associated with one condition.

**Details:** This can lead to critical code executing in situations where it should not.more details.

**Remediation:** Omitting a break statement so that one may fall through is often indistinguishable from an error, and therefore should be avoided. If you need to use fall-through capabilities, make sure that you have clearly documented this within the switch statement, and ensure that you have examined all the logical possibilities.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681549<br>Missing break in switch | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_ojpeg.c:1905 | Other |

## Technical Impact: Gain privileges

### CWE 330: Use of Insufficiently Random Values

**Summary:** The software may use insufficiently random numbers or values in a security context that depends on unpredictable numbers.

**Details:** When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.more details.

**Remediation:** Use a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations with adequate length seeds.

| Issue ID (CID) and Issue Type | Source File and Line Number | Component |
|---|---|---|
| 681625<br>CERT-CPP Miscellaneous | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_luv.c:825 | Other |
| 681561<br>CERT-CPP Miscellaneous | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/liblept/src/utils.c:3188 | Other |
| 681471<br>CERT-CPP Miscellaneous | /OpenAPLR-3.1.1_Vs2022/libs/dependencies/libtiff/src/tif_luv.c:1050 | Other |

**Showing 200 of 4,201 issues with valid CWE entries.**

## Methodology

### Introduction

This report is a distillation of the output of the Coverity Code Advisor used on a particular code source base. Coverity Code Advisor is a static analysis tool that is capable of finding quality defects, security vulnerabilities, and test violations through the process of scanning the output of a specially-compiled code base. The information in this report is specific to security vulnerabilities detected by Coverity Code Advisor and their categorization in the OWASP and CWE/SANS ranking systems.

### About Static Analysis

Static analysis is the analysis of software code without executing the compiled program, for the purpose of finding logic errors or security vulnerabilities. Coverity's static analysis tools integrate with all major build systems and generate a high fidelity representation of source code to provide full code path coverage, ensuring that every line of code and execution path is analyzed. Code Advisor supports the market-leading compilers for C, C++, Java, C#, Objective C, and Javascript.

### About CWE

CWE (Common Weakness Enumeration) is a software community project that is responsible for creating a catalog of software weaknesses and vulnerabilities and is sponsored by the office of Cybersecurity and Communications at the U.S. Department of Homeland Security. The Common Weakness Scoring System (CWSS) provides a method by which to identify and compare weaknesses.

CWE is used by vulnerability-listing efforts such as CWE/SANS Top 25 and OWASP Top 10, among others, to create generalized lists of ranked vulnerabilities. Some, but not all, of the issues reported by Coverity are mapped to CWE-listed vulnerabilities. The Common Weakness Risk Assessment Framework (CWRAF) is a methodology for prioritizing software weaknesses in the context of the software's use. A CWRAF "severity mapping" prioritizes issues according to their CWE technical impact values. There are 8 technical impacts:

1. modify data,
2. read data,
3. create a denial-of-service that results in unreliable execution,
4. create a denial-of-service that results in resource consumption,
5. execute unauthorized code or commands,
6. gain privileges or assume identity,
7. bypass protection mechanism,
8. hide activities

CWRAF and CWSS allow users to rank classes of weaknesses independently of any particular software package, in order to prioritize them relative to each other.

### Setting Priorities with Severity Mappings

A severity mapping is a mapping that determines a severity level, or score, for a given technical impact associated with a software issue. This score can in turn be used to derive the priority assigned to the remediation of the issue. Coverity provides built-in severity mappings to help customers to set these priorities for particular types of applications, and the ability to create custom severity mappings.

The part of the severity mapping that's relevant for this work is the Technical Impact Scorecard. It maps a technical impact to a severity value between Informational (the lowest) and Very High (the highest). This value is known variously as the technical impact's "score" or its "severity". This document uses "severity".

### Scoring Methodology

An issue from Coverity's code analysis will contribute to the security score when it has a CWE ID where the CWE ID maps to at least one of the eight technical impact values, at least one the mapped technical impact values has a severity level greater than Informational, and the issue has not been marked as "False Positive" or "Intentional". A severity mapping determines the mapping of technical impact values to severity levels and it is an issue's assigned severity level that is used for the security score calculation. For an issue where its CWE ID maps to more than one of the eight technical impact values, a single technical impact value will be assigned to the issue, where the highest relevant severity level will determine which technical impact value gets assigned, with ties for the highest severity level being broken arbitrarily.

The severity levels from the Security Details section are used to determine the security score, with the possible severity levels being Very High, High, Medium, Low, and Very Low. The highest severity level that has at least one issue associated with it will greatly influence the security score. Additional issues with the highest severity level will have a greater impact on reducing the security score than will additional issues with a relatively lower severity level. As such, it's important to address issues with the highest severity level.

While the full range of a possible security score is from 0 to 100, with 100 being the best value possible, only a project with a highest severity level of Very Low that contains 6 or less Very Low severity level issues can receive a score of 100. A project would need to contain more than 30000 Very High severity level issues to receive a score lower than 30. Meanwhile, a project with a highest severity level of Very Low would need to contain more than 30000 Very Low severity level issues to receive a score lower than 70.

To give some further context, consider the standard Target Assurance Levels plus their corresponding Target Security Score values of AL1 (90), AL2 (80), AL3 (70), and AL4 (60) relative to the highest severity level that has at least one issue associated with it.

- If Very High severity level issues exist, it will be nearly impossible to achieve AL3 (70), and it will be quite a challenge to achieve AL4 (60).
- If all of the Very High severity level issues have been addressed, but at least one High severity level issue exists, it will be nearly impossible to achieve AL2 (80), and it will be a reasonable challenge to achieve AL3 (70), with AL4 (60) being within easier reach.
- If all of the Very High and High severity level issues have been addressed, but at least one Medium severity level issue exists, it will be nearly impossible to achieve AL1 (90) and quite challenging to achieve AL2 (80), while AL3 (70) is more likely to be within reach, and AL4 (60) should be a relatively easy target to reach.

## The 2017 OWASP Top 10 List

The OWASP (Open Web Application Security Project) Foundation is an international organization whose mission is to advance the cause of secure software. As part of its activities, OWASP publishes a report of the most critical web application security flaws in rank order based on the input of a worldwide group of security experts. The most recent version of this list and accompanying report is the OWASP Top 10 List for 2017. The OWSAP Top 10 List is referenced by many standards including MITRE, PCI DSS, DISA, and the FTC.

## The CWE/SANS Top 25 List

The SANS Institute is a cooperative research and education organization made up security experts from around the world. SANS is a major source of information on computer security and makes available an extensive collection of research documentation. It also operates the Internet’s early security vulnerability warning system, the Internet Storm Center. The 2019 CWE/SANS Top 25 Most Dangerous Software Errors is a list of the most common and critical errors that can lead to software vulnerabilities, as published by this organization.

## About Coverity

Coverity is a leading provider of quality and security testing solutions. The company, founded in the Computer Science Laboratory at Stanford University, provides an array of tools that assist developers in addressing critical quality and security issues early in the development cycle, thus saving development organizations from remediating issues late in the development cycle or after release when they are much more costly.  Many major software development organizations, including 8 of the top 10 global brands and 9 of the top 10 software companies, deploy Coverity analysis tools. Coverity also maintains a free, cloud based analysis platform, called Scan, for the Open Source Community.