Coverity® **Security Report**

ALPR
v. v0701

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 2, 2022 9:52 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#p10079) in order to see source code annotated with remediation recommendations.

Lines of Code Inspected: 2179219

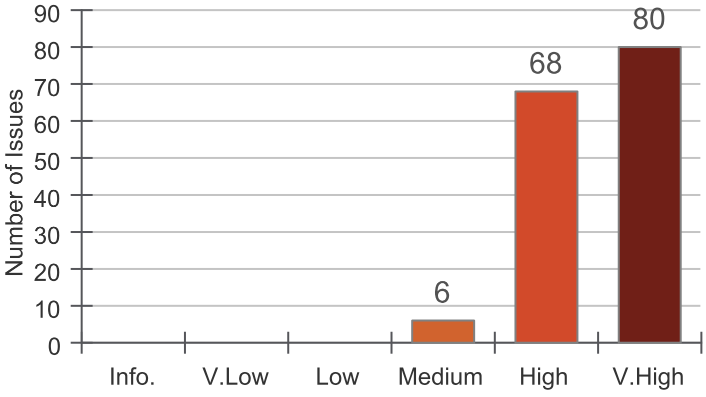
Scorecard

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	49	No
OWASP Top 10 Count	0	0	Yes
CWE/SANS Top 25 Count	0	65	No
Analysis Date	2022-6-2	2022-7-1	Yes
Overall Status			**No**

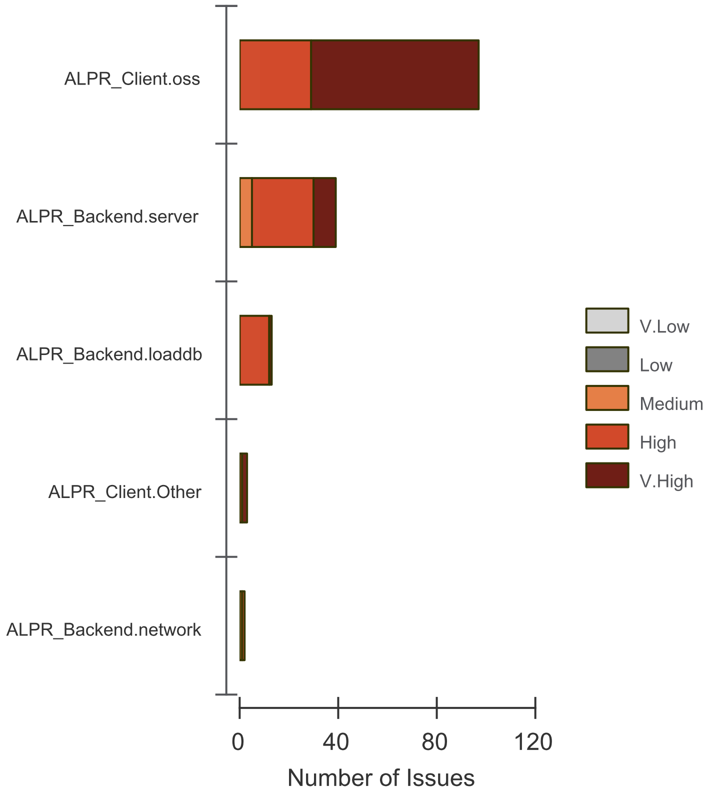
Issues By Severity

A total of 154 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	144
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 49 to 90:

- 80 “Very high” issues.
- 68 “High” issues.
- 4 “Medium” issues.

OWASP Top 10 Remediation

The project has no issues in the OWASP Top 10.

CWE/SANS Top 25 Remediation

Resolve 65 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	80
Gain privileges	Very high	0
Bypass protection mechanism	High	0
Denial of service, unreliable execution	High	66
Modify data	High	2
Denial of service, Resource consumption	Medium	4
Hide activities	Medium	0
Read data	Medium	2
Total		**154**

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_Backend	10453	2022-7-1 9:57:46 오후	2022.3.3	
ALPR_Client	10454	2022-7-1 11:33:2 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	7
2. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	CWE-79	0
3. Improper Input Validation	CWE-20	1
4. Information Exposure	CWE-200	0
5. Out-of-bounds Read	CWE-125	2
6. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	CWE-89	0
7. Use After Free	CWE-416	0
8. Integer Overflow or Wraparound	CWE-190	0
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	55
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	0
21. Missing Release of Resource after Effective Lifetime	CWE-772	0
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		**65**

Detailed Issues Ranked By Severity

Showing 154 of 154 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
675587 CERT-CPP Containers	/OpenALPR/plateServer/server/server.cpp:262	ALPR_Backend.server
675544 CERT-CPP Containers	/OpenALPR/plateServer/server/server.cpp:154	ALPR_Backend.server
675566 CERT-CPP Containers	/OpenALPR/plateServer/server/server.cpp:197	ALPR_Backend.server
675534 CERT-CPP Containers	/OpenALPR/plateServer/server/server.cpp:153	ALPR_Backend.server
675565 CERT-CPP Containers	/OpenALPR/plateServer/server/server.cpp:393	ALPR_Backend.server

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
675536 CERT-CPP Characters and Strings	/OpenALPR/plateServer/server/server.cpp:153	ALPR_Backend.server
675559 CERT-CPP Characters and Strings	/OpenALPR/plateServer/server/server.cpp:394	ALPR_Backend.server

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
675580 CERT-CPP Characters and Strings	/OpenALPR/plateServer/server/ConfigParser.cpp:33	ALPR_Backend.server

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
675819 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/trial/test/test_pyunitcompat.py:217	ALPR_Client.oss
675818 Bad use of null-like value	/OpenALPR/studio/Lib/urllib3/test/test_ssltransport.py:30	ALPR_Client.oss
675815 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:948	ALPR_Client.oss
675812 Bad use of null-like value	/OpenALPR/studio/Lib/django/django/db/models/sql/compiler.py:1258	ALPR_Client.oss
675807 Bad use of null-like value	/OpenALPR/studio/Lib/attrs/tests/test_validators.py:810	ALPR_Client.oss
675799 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ast_transforms.py:161	ALPR_Client.oss
675797 Bad use of null-like value	/OpenALPR/studio/Lib/cryptography/tests/utils.py:789	ALPR_Client.oss
675798 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/cryptography/tests/x509/test_x509_ext.py:1700	ALPR_Client.oss
675796 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1/pyasn1/codec/ber/decoder.py:1695	ALPR_Client.oss
675793 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/trial/test/test_util.py:474	ALPR_Client.oss
675792 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/polynomial/tests/test_printing.py:516	ALPR_Client.oss

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
675786 Bad use of null-like value	/OpenALPR/studio/Lib/django/django/contrib/auth/__init__.py:157	ALPR_Client.oss
675785 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/lib/histograms.py:853	ALPR_Client.oss
675783 Bad use of null-like value	/OpenALPR/studio/Lib/autobahn-python/autobahn/wamp/protocol.py:1656	ALPR_Client.oss
675784 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/twisted/src/twisted/internet/defer.py:1746	ALPR_Client.oss
675781 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:901	ALPR_Client.oss
675779 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1/pyasn1/codec/ber/encoder.py:800	ALPR_Client.oss
675777 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/test/test_failure.py:96	ALPR_Client.oss
675670 Dereference before null check	/OpenALPR/plateServer/loaddb/loaddb.cpp:75	ALPR_Backend.loaddb
675771 Bad use of null-like value	/OpenALPR/studio/Lib/pyopenssl/tests/test_ssl.py:2337	ALPR_Client.oss
675768 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/mail/pop3.py:155	ALPR_Client.oss
675765 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/webapp/alpr/views.py:87	ALPR_Client.Other
675766 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/urllib3/src/urllib3/response.py:950	ALPR_Client.oss
675545 CERT-CPP Characters and Strings	/OpenALPR/plateServer/server/plog/Util.h:322	ALPR_Backend.server
675715 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/core/numeric.py:1654	ALPR_Client.oss
675710 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1/pyasn1/codec/ber/decoder.py:626	ALPR_Client.oss
675709 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/txaio/test/_asyncio_test_utils.py:112	ALPR_Client.oss
675761 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:1193	ALPR_Client.oss
675760 Bad use of null-like value	/OpenALPR/studio/webapp/alpr/views.py:208	ALPR_Client.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
675759 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/core/numeric.py:1663	ALPR_Client.oss
675758 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/test/test_failure.py:892	ALPR_Client.oss
675757 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/test/test_failure.py:434	ALPR_Client.oss
675756 Bad use of null-like value	/OpenALPR/studio/Lib/cryptography/tests/utils.py:753	ALPR_Client.oss
675755 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1-modules/tests/test_rfc5752.py:116	ALPR_Client.oss
675751 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/distutils/command/build_clib.py:411	ALPR_Client.oss
675750 Bad use of null-like value	/OpenALPR/studio/Lib/autobahn-python/autobahn/xbr/_seller.py:715	ALPR_Client.oss
675747 Bad use of null-like value	/OpenALPR/studio/Lib/numpy/numpy/core/numeric.py:1666	ALPR_Client.oss
675746 Bad use of null-like value	/OpenALPR/studio/Lib/urllib3/test/test_ssltransport.py:25	ALPR_Client.oss
675745 Bad use of null-like value	/OpenALPR/studio/Lib/autobahn-python/autobahn/xbr/_seller.py:567	ALPR_Client.oss
675743 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:1240	ALPR_Client.oss
675742 Bad use of null-like value	/OpenALPR/studio/Lib/cryptography/tests/utils.py:743	ALPR_Client.oss
675740 Bad use of null-like value	/OpenALPR/studio/Lib/asgiref/asgiref/sync.py:292	ALPR_Client.oss
675736 Bad use of null-like value	/OpenALPR/studio/Lib/autobahn-python/autobahn/wamp/protocol.py:1453	ALPR_Client.oss
675735 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/twisted/src/twisted/internet/defer.py:1755	ALPR_Client.oss
675734 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1/pyasn1/codec/ber/decoder.py:1755	ALPR_Client.oss
675731 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/runner/inetdconf.py:63	ALPR_Client.oss
675729 Bad use of null-like value	/OpenALPR/studio/Lib/cryptography/tests/utils.py:863	ALPR_Client.oss
675727 Bad use of null-like value	/OpenALPR/studio/Lib/cryptography/tests/utils.py:749	ALPR_Client.oss

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
675726 Bad use of null-like value	/OpenALPR/studio/Lib/pycparser/pycparser/ast_transforms.py:161	ALPR_Client.oss
675722 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/test/test_failure.py:879	ALPR_Client.oss
675721 Bad use of null-like value	/OpenALPR/studio/Lib/twisted/src/twisted/test/test_failure.py:434	ALPR_Client.oss
675720 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/autobahn-python/autobahn/twisted/test/test_tx_component.py:241	ALPR_Client.oss
675714 Bad use of null-like value	/OpenALPR/studio/Lib/pyasn1/pyasn1/codec/native/decoder.py:150	ALPR_Client.oss
675712 Bad use of null-like value	/OpenALPR/studio/Lib/requests/requests/sessions.py:761	ALPR_Client.oss
675713 Attribute/item access or function call before check for None or undefined	/OpenALPR/studio/Lib/numpy/numpy/ma/tests/test_core.py:1817	ALPR_Client.oss

CWE 569: Expression Issues

Summary: Weaknesses in this category are related to incorrectly written expressions within code.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675813 Operands don't affect result	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_interface.py:1060	ALPR_Client.oss
675811 Bitwise-and with zero	/OpenALPR/studio/Lib/pyasn1/tests/type/test_univ.py:169	ALPR_Client.oss
675801 Same on both sides	/OpenALPR/studio/Lib/twisted/src/twisted/internet/test/test_base.py:276	ALPR_Client.oss
675794 Operands don't affect result	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_declarations.py:456	ALPR_Client.oss
675791 Same on both sides	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_registry.py:2603	ALPR_Client.oss
675790 Same on both sides	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_registry.py:2812	ALPR_Client.oss
675789 Same on both sides	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_registry.py:2774	ALPR_Client.oss
675763 Same on both sides	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_interface.py:1070	ALPR_Client.oss

Technical Impact: Execute unauthorized code

CWE 569: Expression Issues

Summary: Weaknesses in this category are related to incorrectly written expressions within code.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675754 Same on both sides	/OpenALPR/studio/Lib/numpy/numpy/matrixlib/tests/test_defmatrix.py:158	ALPR_Client.oss
675749 Bitwise-and with zero	/OpenALPR/studio/Lib/pyasn1/tests/type/test_univ.py:160	ALPR_Client.oss
675739 Same on both sides	/OpenALPR/studio/Lib/numpy/numpy/matrixlib/tests/test_defmatrix.py:163	ALPR_Client.oss
675733 Same on both sides	/OpenALPR/studio/Lib/twisted/src/twisted/internet/test/test_base.py:299	ALPR_Client.oss
675732 Same on both sides	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_registry.py:2565	ALPR_Client.oss
675730 Same on both sides	/OpenALPR/studio/Lib/twisted/src/twisted/internet/test/test_base.py:298	ALPR_Client.oss
675725 Same on both sides	/OpenALPR/studio/Lib/attrs/tests/test_make.py:1761	ALPR_Client.oss
675719 Operands don't affect result	/OpenALPR/studio/Lib/numpy/numpy/core/tests/test_numeric.py:425	ALPR_Client.oss
675717 Same on both sides	/OpenALPR/studio/Lib/twisted/src/twisted/internet/test/test_base.py:275	ALPR_Client.oss

Severity: High

Technical Impact: Denial of service, unreliable execution

CWE 248: Uncaught Exception

Summary: An exception is thrown from a function, but it is not caught.

Details: When an exception is not caught, it may cause the program to crash or expose sensitive information.more details.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675604 Uncaught exception	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675653 Uncaught exception	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675630 Uncaught exception	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675627 Uncaught exception	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675623 Uncaught exception	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675592 Uncaught exception	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server

Technical Impact: Denial of service, unreliable execution

CWE 561: Dead Code

Summary: The software contains dead code, which can never be executed.

Details: Dead code is source code that can never be executed in a running program. The surrounding code makes it impossible for a section of code to ever be executed.more details.

Remediation: Remove dead code before deploying the application.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675816 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:744	ALPR_Client.oss
675810 Logically dead code	/OpenALPR/studio/Lib/numpy/numpy/matrixlib/defmatrix.py:159	ALPR_Client.oss
675809 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:380	ALPR_Client.oss
675806 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:281	ALPR_Client.oss
675805 Logically dead code	/OpenALPR/studio/webapp/alpr/views.py:91	ALPR_Client.Other
675804 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:394	ALPR_Client.oss
675803 Structurally dead code	/OpenALPR/studio/Lib/twisted/docs/historic/2003/pycon/deferex/deferex-complex-raise.py:8	ALPR_Client.oss
675802 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:1122	ALPR_Client.oss
675800 Logically dead code	/OpenALPR/studio/Lib/autobahn-python/autobahn/wamp/message.py:284	ALPR_Client.oss
675795 Structurally dead code	/OpenALPR/studio/Lib/twisted/docs/historic/2003/pycon/deferex/deferex-simple-raise.py:3	ALPR_Client.oss
675787 Structurally dead code	/OpenALPR/studio/Lib/autobahn-python/examples/asciinema-autobahn-demo.py:30	ALPR_Client.oss
675782 Logically dead code	/OpenALPR/studio/Lib/daphne/tests/test_websocket.py:50	ALPR_Client.oss
675778 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:712	ALPR_Client.oss
675776 Logically dead code	/OpenALPR/studio/Lib/autobahn-python/autobahn/asyncio/rawsocket.py:164	ALPR_Client.oss
675775 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:509	ALPR_Client.oss
675773 Logically dead code	/OpenALPR/studio/Lib/requests/requests/models.py:153	ALPR_Client.oss
675770 Logically dead code	/OpenALPR/studio/Lib/twisted/src/twisted/protocols/loopback.py:205	ALPR_Client.oss
675769 Logically dead code	/OpenALPR/studio/Lib/daphne/tests/test_http_request.py:55	ALPR_Client.oss
675708 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:495	ALPR_Client.oss
675762 Structurally dead code	/OpenALPR/studio/Lib/pycparser/pycparser/c_ast.py:728	ALPR_Client.oss

Technical Impact: Denial of service, unreliable execution

CWE 561: Dead Code

Summary: The software contains dead code, which can never be executed.

Details: Dead code is source code that can never be executed in a running program. The surrounding code makes it impossible for a section of code to ever be executed.more details.

Remediation: Remove dead code before deploying the application.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675748 Logically dead code	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:2783	ALPR_Client.oss
675744 Logically dead code	/OpenALPR/studio/Lib/pycparser/pycparser/ply/lex.py:597	ALPR_Client.oss
675738 Structurally dead code	/OpenALPR/studio/Lib/attrs/src/attr/_compat.py:101	ALPR_Client.oss
675728 Logically dead code	/OpenALPR/studio/Lib/pycparser/pycparser/ply/yacc.py:3071	ALPR_Client.oss
675711 Logically dead code	/OpenALPR/studio/Lib/requests/requests/auth.py:230	ALPR_Client.oss

CWE 686: Function Call With Incorrect Argument Type

Summary: The software calls a function, procedure, or routine, but the caller specifies an argument that is the wrong data type, which may lead to resultant weaknesses.

Details: This weakness is most likely to occur in loosely typed languages, or in strongly typed languages in which the types of variable arguments cannot be enforced at compilation time, or where there is implicit casting.more details.

Remediation: Because this function call often produces incorrect behavior it will usually be detected during testing or normal operation of the software. During testing exercise all possible control paths will typically expose this weakness except in rare cases when the incorrect function call accidentally produces the correct results or if the provided argument type is very similar to the expected argument type.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675673 Invalid type in argument to printf format specifier	/OpenALPR/plateServer/loaddb/loaddb.cpp:60	ALPR_Backend.loaddb
675661 Invalid type in argument to printf format specifier	/OpenALPR/plateServer/common/NetworkTCP.cpp:380	ALPR_Backend.network
675647 Invalid type in argument to printf format specifier	/OpenALPR/plateServer/server/server.cpp:492	ALPR_Backend.server

Technical Impact: Denial of service, unreliable execution

CWE 688: Function Call With Incorrect Variable or Reference as Argument

Summary: The software calls a function, procedure, or routine, but the caller specifies the wrong variable or reference as one of the arguments, which may lead to undefined behavior and resultant weaknesses.

Remediation: Because this function call often produces incorrect behavior it will usually be detected during testing or normal operation of the software. During testing exercise all possible control paths will typically expose this weakness except in rare cases when the incorrect function call accidentally produces the correct results or if the provided argument type is very similar to the expected argument type.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675817 Typo in identifier	/OpenALPR/studio/Lib/twisted/src/twisted/names/secondary.py:166	ALPR_Client.oss
675767 Typo in identifier	/OpenALPR/studio/Lib/twisted/src/twisted/plugin.py:227	ALPR_Client.oss
675764 Typo in identifier	/OpenALPR/studio/Lib/zope.interface/src/zope/interface/tests/test_advice.py:116	ALPR_Client.oss
675753 Typo in identifier	/OpenALPR/studio/Lib/twisted/src/twisted/mail/scripts/mailmail.py:132	ALPR_Client.oss
675737 Typo in identifier	/OpenALPR/studio/Lib/twisted/src/twisted/mail/pb.py:64	ALPR_Client.oss

CWE 703: Improper Check or Handling of Exceptional Conditions

Summary: The software does not properly anticipate or handle exceptional conditions that rarely occur during normal operation of the software.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675573 CERT-CPP Memory Management	/OpenALPR/plateServer/server/plog/Record.h:317	ALPR_Backend.server

CWE 754: Improper Check for Unusual or Exceptional Conditions

Summary: The software does not check or improperly checks for unusual or exceptional conditions that are not expected to occur frequently during day to day operation of the software.

Details: The programmer may assume that certain events or conditions will never occur or do not need to be worried about, such as low memory conditions, lack of access to resources due to restrictive permissions, or misbehaving clients or components. However, attackers may intentionally trigger these unusual conditions, thus violating the programmer's assumptions, possibly introducing instability, incorrect behavior, or a vulnerability.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
675589 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server

Technical Impact: Denial of service, unreliable execution

CWE 754: Improper Check for Unusual or Exceptional Conditions

Summary: The software does not check or improperly checks for unusual or exceptional conditions that are not expected to occur frequently during day to day operation of the software.

Details: The programmer may assume that certain events or conditions will never occur or do not need to be worried about, such as low memory conditions, lack of access to resources due to restrictive permissions, or misbehaving clients or components. However, attackers may intentionally trigger these unusual conditions, thus violating the programmer's assumptions, possibly introducing instability, incorrect behavior, or a vulnerability.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
675585 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675556 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675554 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675579 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675578 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675571 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675572 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675550 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675549 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675547 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675543 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675569 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675567 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server

Technical Impact: Denial of service, unreliable execution

CWE 754: Improper Check for Unusual or Exceptional Conditions

Summary: The software does not check or improperly checks for unusual or exceptional conditions that are not expected to occur frequently during day to day operation of the software.

Details: The programmer may assume that certain events or conditions will never occur or do not need to be worried about, such as low memory conditions, lack of access to resources due to restrictive permissions, or misbehaving clients or components. However, attackers may intentionally trigger these unusual conditions, thus violating the programmer's assumptions, possibly introducing instability, incorrect behavior, or a vulnerability.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
675540 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675538 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675537 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675523 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675562 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675563 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675561 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675531 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675532 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675526 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server
675525 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/loaddb/loaddb.cpp:12	ALPR_Backend.loaddb
675524 CERT-CPP Exceptions and Error Handling	/OpenALPR/plateServer/server/server.cpp:73	ALPR_Backend.server

Technical Impact: Modify data

CWE 459: Incomplete Cleanup

Summary: The software does not properly "clean up" and remove temporary or supporting resources after they have been used.

Remediation: Temporary files and other supporting resources should be deleted/released immediately after they are no longer needed.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675586 CERT-CPP Input/Output	/OpenALPR/plateServer/loaddb/loaddb.cpp:55	ALPR_Backend.loaddb
675583 CERT-CPP Input/Output	/OpenALPR/plateServer/server/ConfigParser.cpp:13	ALPR_Backend.server

Severity: Medium

Technical Impact: Denial of service, Resource consumption

CWE 404: Improper Resource Shutdown or Release

Summary: The program does not release or incorrectly releases a resource before it is made available for re-use.

Details: When a resource is created or allocated, the developer is responsible for properly releasing the resource as well as accounting for all potential paths of expiration or invalidation, such as a set period of time or revocation.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
675700 Resource leak on an exceptional path	/OpenALPR/plateServer/server/plog/Record.h:317	ALPR_Backend.server
675696 Resource leak	/OpenALPR/plateServer/common/NetworkTCP.cpp:234	ALPR_Backend.network
675652 Resource leak	/OpenALPR/plateServer/server/server.cpp:586	ALPR_Backend.server

Technical Impact: Denial of service, Resource consumption

675615 Resource leak	/OpenALPR/plateServer/server/server.cpp:651	ALPR_Backend.server

Technical Impact: Read data

CWE 125: Out-of-bounds Read

This CWE entry is at position 5 in the CWE/SANS Top 25.

Summary: The software reads data past the end, or before the beginning, of the intended buffer.

Details: Typically, this can allow attackers to read sensitive information from other memory locations or cause a crash. A crash can occur when the code reads a variable amount of data and assumes that a sentinel exists to stop the read operation, such as a NUL in a string. The expected sentinel might not be located in the out-of-bounds memory, causinfg excessive data to be read, leading to a segmentation fault or a buffer overflow. 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 read operation then produces undefined or unexpected results. more details.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
675688 Out-of-bounds read	/OpenALPR/plateServer/server/server.cpp:541	ALPR_Backend.server
675687 Out-of-bounds read	/OpenALPR/plateServer/server/server.cpp:256	ALPR_Backend.server

Showing 154 of 154 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.