Coverity Scan:

2011 Open Source Integrity Report

Table of Contents

The Development Testing Imperative

The importance of software as a source of competitive advantage only continues to rise from year to year, while development cycles continue to shrink. The ongoing challenge of developing and maintaining high quality software while responding to rapid market demands for new features and functionality is compounded with increasing software complexity—from the increasing size of codebases to the number of suppliers contributing to the modern software supply chain. As part of this software supply chain, open source code continues to be an integral part of mainstream software development projects. In January 2011, analyst firm Gartner[1] estimated that by 2016, open source software will be included in mission-critical software portfolios within 99% of Global 2000 enterprises—up from 75% in 2010. But as the lines continue to blur between proprietary and open source code as part of commercial software development, development organizations face significant challenges. They must efficiently and effectively gain visibility into and control over the quality of open source code being integrated into the project to ensure it's at the right quality level and meets the pre-defined acceptance criteria, both as a standalone software component and when integrated and operating together with other software components.

These challenges have prompted development organizations across a variety of industries to adopt new processes and technologies to better meet business demands without impacting development time and cost. Development testing has emerged as a new category that includes a set of processes and software, such as static analysis, designed to help development organizations easily find and fix software problems early in the development cycle, as the code is being written, without impacting time to market, cost and customer satisfaction. Development testing augments traditional testing, including QA functional and performance testing, as well as security audits. This provides development teams with a way to test their code for defects in a non-intrusive manner, so development stays focused on innovation, management gets visibility into problems early in the cycle to make better decisions, and the business continues to deliver high-quality products to market for competitive advantage.

Our mission with development testing is to empower software development, from commercial organizations to open source software (OSS) projects, regardless of industry or size, to seamlessly build quality into existing development processes. We work with over 1,100 customers to provide a platform for automated code testing and policy management across both proprietary and open source software. This includes defining standard SLAs for code quality, security, and complexity, testing against those policies for software defects, and providing an objective and ongoing measurement of quality over time. This gives organizations visibility into both quality improvement and quality risk. As we continue to enhance our development testing platform for commercial software development projects, we offer this technology as a service to the open source community via Coverity Scan.

For those readers new to Coverity Scan, it is the largest public-private sector research project in the world focused on open source integrity, originally initiated in 2006 with the U.S. Department of Homeland Security. Today, Coverity owns and manages the Scan program, and has worked with over 300 of the most widely adopted open source projects over the past five years—including Linux, PHP, Apache, Firefox, and Android—to automatically scan, or test, their software code during development. The foundation of our development testing platform and the technology that powers Coverity Scan is our core analysis technology, Coverity® Static Analysis.

[1] Gartner: A CIO's Perspective on Open-Source Software, January 2011

Our mission with Coverity Scan is to provide open source projects that are committed to quality with an opportunity to build quality into their process in the same way as commercial projects. We are continually learning from our commercial customers and building this knowledge into our technology—from improved accuracy to ease-of-use to fit into the development workflow—in order to bring these best practices to the open source community. In addition, we are seeing more commercial projects build policy management and governance into their development processes. By defining standard code quality and security policies for software components based upon business criticality—whether internally developed or third-party provided—commercial projects can create a common and objective baseline to measure code quality against to ensure the right SLAs are upheld. To meet this increasing demand for visibility, we aim to provide an objective measurement of open source quality to increase its visibility within commercial projects and drive open source adoption. Ultimately, Coverity Scan is focused on the continuous improvement of code quality across both commercial and open source software development.

Coverity Scan Evolution

Since Coverity Scan was initiated in 2006, static analysis as a technology has significantly evolved and matured, along with its adoption within the software development community. New capabilities have been added and analysis engines have become more sophisticated, resulting in improved detection rates for both new and existing types of defects in software code. Technological advances, coupled with ease of use enhancements and better integration into existing development workflows, have improved overall developer adoption of static analysis.

In the 2010 Coverity Scan Open Source Integrity Report, we discussed a few upcoming changes to Coverity Scan, including an upgrade of active projects to the latest version of Coverity software to bring our technology advances to the open source community. At the beginning of 2011, the open source projects in Coverity Scan were analyzed using a mix of versions of Coverity Static Analysis. We have been working hard over the past year to upgrade all active projects to the Coverity 5 development testing platform analysis engine. This has allowed us to surface and bring more visibility to software defects in open source projects to further improve quality, both in terms of the quantity of existing defects detected and new defect categories. With the upgrade to Coverity 5, many new checkers have been added to the analysis engine, many existing checkers have been improved to find more potential defects, and the checkers have been reorganized into categories to help developers prioritize fixes.

Also, a subset of projects highlighted in this year's report have been upgraded to version 5.5 of Coverity Integrity Manager (CIM). Besides adding a robust web services interface which allows seamless integration with build, source control management, and bug tracking systems, this enhanced defect management interface makes it easy for developers to understand the severity and potential impact of defects that may occur in code shared by multiple products or development branches. CIM allows developers to leverage a rich defect knowledge base through references such as links to the Common Weakness Enumeration (CWE) for each defect. As a by-product of this new data model, CIM 5.5 counts defects differently from previous versions of the product, and therefore the statistics for outstanding defects are not an "apples to apples" comparison to those reported in previous years' Scan reports. In addition to upgrades in the underlying technology that powers Coverity Scan, in 2011 we added on-demand analysis capabilities which allow projects to schedule analysis at times that fit best within their development process, rather than running Coverity on an arbitrary schedule. The Coverity build is now run as part of the project's build automation system. Coverity runs the analysis virtually in the cloud, and makes the results available via CIM. This integration enables more seamless workflow integration by moving the control over the build to the open source project, and has driven further adoption from open source projects.

Over the course of 2012, we will continue to upgrade all open source projects in Coverity Scan to the Coverity 5 platform to take advantage of advances in analysis maturity and to leverage CIM's defect management interface so developers can customize the defect management workflow in a way that best fits their project.

In addition, in last year's report we discussed providing ongoing visibility to open source quality by publishing an automatically generated Coverity Software Integrity Report (Integrity Report) on a variety of open source projects. Last year's Scan report included an Integrity Report for the Android Kernel 2.6.32 ("Froyo). This year's report includes Integrity Reports for three widely adopted open source projects in the industry: Linux 2.6, PHP 5.3, and PostgreSQL 9.1. These projects are model citizens and best practices examples of active Scan projects that have adopted development testing via static analysis as an

automated testing method to help drive and improve quality in open source software over time. These three projects represent a mix of software types, including system software, programming software, and application software, and a mix of codebase sizes ranging from just over 500,000 lines of code to almost 7 million lines of code. This illustrates that regardless of size or type of codebase, committing to quality via the adoption of development testing directly correlates to high quality and continued improvement over time. As commercial projects continue to build policy management and governance into their development process to ensure code delivered across the software supply chain, including open source, meets established acceptance criteria, we will continue to issue Integrity Reports for open source projects in Coverity Scan to bring ongoing visibility to commercial projects.

As open source adoption continues in commercial software development, a commonly asked question is, "How does open source software stack up against proprietary software?" This year's report marks the first time we are making a comparison between the quality of open source software in a sample of active projects in Scan and a representative sample of proprietary codebases from anonymous Coverity users across a variety of industries. What we found is that when comparing codebases of similar size, quality is on par across open source and proprietary software. This validates that projects that are committed to quality and adopt processes and technologies such as development testing—whether it's in the open source community or within commercial software development in the enterprise—will reap the benefits of ongoing quality improvement over time.

Finally, this year we have appointed Zack Samocha as the Scan Project Director for Coverity. Samocha spent nearly a decade at Mercury Interactive, now Hewlett-Packard, where he was instrumental in the development of its quality assurance testing products, as well as building quality best practices within enterprise application development. His enterprise development and testing experience will be instrumental in bringing development testing best practices to the open source community.

The State of Open Source Software Integrity

With each year's report we are asked to share our insight into the state of open source software integrity. Due to the improvements made to Coverity Scan and its underlying technology over the past year, which we discussed in the previous section, we were able to flag more defects than in prior years and therefore cannot make direct comparisons to prior years' data. This year we analyzed over 37 million lines of code from 45 of the most active projects in Coverity Scan, 32 of which were active prior to 2011 and many of which have been in Coverity Scan since 2006 at its inception. This gave us a baseline of quality against industry average. To keep the results most relevant, we looked only at high- and medium-impact defects for all our calculations.

While the average codebase size for the open source projects in our analysis is 832,000 lines of code, the range of codebases varied from under 100,000 lines of code to two projects with codebases totaling nearly 7 million lines of code. While all projects had active developer support and participation, support ranged from a single user to nearly 100 active users.

TABLE A: SAMPLE PROJECT DISTRIBUTION BY CODEBASE SIZE

Size of Codebase (Lines of Code)	Number of Projects
Under 100,000	10
100,000 – 500,000	20
500,000 – 1 million	7
1 million – 4 million	6
Over 7 million	2

This year's analysis led us to our first key finding:

Open source quality for active projects in Coverity Scan is better than the software industry average.

Based upon the sample of active projects in Coverity Scan, we found the quality of open source software is above average. This is based upon a measure of defect density against the industry average defect density for the software industry. The average defect density, or the number of defects per thousand lines of code, across the top 45 active open source projects in Scan is .45. Coverity's experience working with commercial software development projects has shown that a good benchmark for high-quality software is an average defect density of equal to or less than 1.0. As discussed, many of the projects included in this year's analysis have been active in Scan since its inception in 2006, and this remarkably low average defect density is a marker of the mature rate of adoption of static analysis within the open source community as a testing measure to help open source developers fix defects and improve overall software quality. See Appendix A for additional detail on the aggregate findings.

A note on outstanding defects: The total number of outstanding defects at the end of 2011 should not be viewed as an indicator that quality is declining. As discussed, we added new analysis capabilities this year which resulted in an increase in this number over the course of the year. It should also be noted that open source developers across all projects fixed 6,133 defects in 2011, an increase over 2010. We will continue to report on these metrics moving forward to track year over year trends, beginning again with the 2012 report. Regarding the most common defect types found, control flow issues top the list of outstanding defects this year. This category includes defects in which the program contains code that either never executes (dead code), or executes under the wrong conditions, and could lead to unexpected behavior or security vulnerabilities. While absent as a defect type from the 2010 report, it should not be interpreted that these defects did not exist last year. It's simply due to new defect categorization as part of the upgraded analysis engine for Coverity Scan projects. Many new defect categories in this year's report are an aggregation of multiple defect types which were each reported as individual defect categories in previous years' reports (see Appendix A for additional detail).

DEFECT DENSITY

In the context of this report, "defect density" refers to static analysis results found by Coverity Static Analysis, not defects found through testing or post-deployment use. Defect density is computed using only defects in the "high-impact" and "medium-impact" categories, and is a measure of confirmed and potential defects that are left in the codebase as of the time of the report. Defect density is computed by dividing the number of defects found by the size of the codebase in lines of code. The advantage of using defect density is that it accounts for the differing size of software code, which makes defect density figures directly comparable among projects of differing sizes. For the Coverity Software Integrity Report, we chose the thresholds for defect density for the integrity levels based on an analysis of data from our customers, prospects, and open source software. We also adjusted the thresholds to round figures to make it simpler to understand and remember the thresholds. We believe that the standards defined in these levels are reasonable, and fairly stringent, standards for software integrity.

Detailed Examination of Linux, PHP and PostgreSQL

Next, we looked at three open source projects in more detail—each well known and widely adopted—that are considered "model citizens" of Coverity Scan. These projects are best-practices examples of how adopting development testing via static analysis can help drive and improve quality in open source software over time. The three projects are Linux 2.6, PHP 5.3, and PostgreSQL 9.1.

Table B: Key Findings for Linux 2.6, PHP 5.3, and PostgreSQL 9.1

	Linux 2.6	PHP 5.3	PostgreSQL 9.1
Lines of code scanned	6,849,378	537,871	1,105,634
Defect Density (as of 12/31/11)	0.62	0.20	0.21
Number of outstanding defects (as of 12/31/11)	4,261	97	233
Number of defects fixed in 2011	1,283	210	78
Number of outstanding defects (as of 1/1/11)	3,457	14	247

This year's analysis led us to our next set of key findings:

> ***Open source projects that build development testing into their process reap the benefits of continued quality improvement over time.***
>
> ***Open source projects of all sizes can successfully adopt development testing, but quality must be measured as a function of codebase size and developer community.***

Linux 2.6, PHP 5.3, and PostgreSQL 9.1 are recognized as open source projects with superior code quality and can be used as industry benchmarks with defect densities of .62, .20, and .21 respectively. These projects all have active, ongoing developer support with Coverity Scan, which illustrates that adoption of development testing via static analysis will help drive and improve quality over time. See Appendices B - D for additional detail via the Coverity Software Integrity Reports for each project.

Linux 2.6

The Linux operating system has widespread adoption in every sector of the information technology industry, from servers to desktops to notebooks to embedded applications. With a defect density of 0.62, it has substantially fewer reported defects than the average for the software industry of 1.0, but more than the .45 average across open source projects sampled in Coverity Scan. However, it is important to note that this is a function of codebase size and the number of people working on it versus a statement of quality. A codebase of half a million lines of code can be effectively managed by a small group of people, requiring

less coordination and effort to fix the majority of defects found by static analysis. Linux version 2.6 grew from 5.3 million lines of code to 6.8 million lines of code between December 2010 and December 2011, and the soon-to-be-released version 3.3 is reportedly 15 million lines of code. With this codebase size, it likely has few people who know enough of the code to feel comfortable triaging and fixing defects throughout the codebase. In addition, as more people are required to be involved in resolving defects it takes longer to fix all of the defects—and particularly in a large codebase. Even harmless defects may require a much larger effort to triage than in a smaller project.

It's also important to look at the software supply chain within Linux itself. Breaking down the defect density within each of the software components, the kernel has a higher defect density. This is likely because every fix has to be weighed against the risk of destabilizing existing code—it's the "some fixes shouldn't be made until you are changing that area of the code" principle. Also, kernel developers may be reluctant to change code that is known from experience to be stable in the field just to satisfy static analysis results. They may wait until the code is being altered for other purposes to incorporate defect fixes into the new code. On the other hand, the kernel has one of the fewest number of defects classified as high risk compared against other components such as device drivers. This is likely due to the criticality and widespread usage of the kernel compared to device drivers, many of which are of interest to only a small portion of the Linux global user base. This may result in a higher tolerance for defects and/or increased effort required in getting the community to commit to inspecting all reported defects over time.

PHP 5.3

PHP, or Hypertext Processor, is an HTML-embedded scripting language. The goal of the language is to allow web developers to write dynamically generated pages quickly. PHP development began almost 20 years ago, and today is installed on millions of web sites and web servers, including some of the world's highest-traffic web sites.

PHP is a smaller codebase in size with just over 500,000 lines of code. This has its advantages for defect management and resolution, as illustrated by its .20 defect density. It is comprehensible to a small, well-coordinated development team, which can feasibly commit to inspecting, understanding, and characterizing every reported potential defect. Thus, a smaller number of developers could fix a greater number of defects in a short amount of time, because it takes fewer people to understand all of the components. PHP started 2011 in Coverity Scan with only 14 defects, proving it's easier to keep up and catch up with a smaller, more easily manageable codebase.

The next step for PHP is to further integrate Coverity Scan results into its test and code coverage analysis via the gcov.php.net site. By integrating Coverity Scan into PHP's existing build, source control management, and bug tracking systems, we are confident that PHP will realize even more quality improvement over time. As stated by Rasmus Lerdorf, creator of PHP:

> "The quality of our code is critical to the ongoing success and adoption of PHP, which includes some of the world's most popular web sites. As our code grows and becomes more complex, Scan will become even more important for us as a way to help improve our code quality."

PostgreSQL 9.1

PostgreSQL is an object-relational database management system, and with over 15 years of active development has earned a strong reputation for reliability, data integrity, and correctness. Today, many organizations, government agencies, and companies use PostgreSQL as the back-end database and data warehouse in their products, including ADP, Cisco, NTT Data, Skype, Research in Motion, The American Chemical Society, and Coverity. Today, it's rare to find a large corporation or government agency that isn't using PostgreSQL in at least one department. Given its widespread adoption and reputation for reliability and integrity, it's no surprise that PostgreSQL has a commitment to code quality. With just over 1 million lines of code and a .21 defect density, PostgreSQL has definitely reaped the benefits of active adoption of static analysis and participation in Coverity Scan.

Development Testing in Action: BRL-CAD Case Study

Part of Coverity Scan's value is active collaboration with the open source community to help them successfully adopt static analysis into their projects. As such, we gain valuable insight into how projects are approaching defect management and resolution. Beginning with this year's report, we will highlight real-world case studies of how projects are implementing Coverity Scan into their workflow, starting with the open source project BRL-CAD.

BRL-CAD is a powerful cross-platform open source combinatorial Constructive Solid Geometry (CSG) solid modeling system. With almost 30 years of active development, BRL-CAD is believed to be the oldest open source codebase in the world that's still under active development, and has been the primary tri-service solid modeling CAD system used by the U.S. military to model weapons systems for vulnerability and lethality analyses. The solid modeling system is frequently used in a wide range of military, academic, and industrial applications including in the design and analysis of vehicles, mechanical parts, and architecture. The package has also been used in radiation dose planning and medical visualization. BRL-CAD is written in a combination of C/C++ code, is just over 1 million lines of code, and includes a developer community of approximately 12 core developers and 20 committers to the source code repository. Due to the critical nature of how and where the code is used, BRL-CAD has a long-standing commitment to code quality and implements strict compliance in the code. The BRL-CAD team has spent a lot of time maintaining and cleaning the codebase, including frequent refactoring, and has implemented a variety of quality measures over the course of development, including Coverity Scan.

In April 2011, a full scan of BRL-CAD source code was put through Coverity Static Analysis as part of its involvement in Coverity Scan, evaluating approximately 840,000 lines of code (1.2 million actual codebase size with comments and whitespace). Over 1,800 defects were flagged by Coverity Scan. According to a project leader, Coverity showed BRL-CAD "a different caliber or class of issue" than other testing and quality measures they have adopted in the past. The project leaders were "quite ecstatic" with the results provided via Coverity Scan, but then needed a way to address these newly found defects. In November 2011, Christopher Sean Morrison, a lead contributor to BRL-CAD, posted a call for participation to the developer community for a physical get-together to share knowledge about defects and help resolve the issues detected. The call for participation can be found here: http://brlcad.org/d/node/87.

The BRL-CAD project community off-site was held in January 2012, just outside Baltimore. Half of the core development team participated in person and two contributed remotely for eight hours a day over a five-day span. The BRL-CAD development team first outlined the entire defect management workflow. They were able to customize Coverity's defect management interface to build their desired workflow, including defect assignments and defect categories. For their defect categorization, BRL-CAD used a combination of four sources: (1) Coverity's defect guidelines for high- and medium-impact defects, (2) the 2009 Air Force Research Laboratory report "The Open Source Hardening Project," (3) the Common Weakness Enumeration, and (4) help from the Coverity Scan team. The group then discussed the process, what is considered a "reviewed" bug, and walked through fixing a couple of issues as a group to test the process.

The goal of the off-site was to fix all outstanding defects instead of a multi-pass "prioritize then fix" approach. Specifically:

- Inspect and fix all issues discovered by Coverity
- Peer review, verify, and validate all changes
- Categorize and document all changes
- Educate on common defect patterns

So how did the team do?

- Inspect and fix all issues discovered by Coverity – 90%
- Peer review, verify, and validate all changes – 10%
- Categorize and document all changes – 100%
- Educate on common defect patterns – 100%

The group was able to decrease outstanding defects from 1,840 to 186 after 5 days, a beginning defect density of .0022 which decreased to .00022. This equated to approximately 330 defect fixes per day, 42 fixes per hour and a fix every 87 seconds across the team. Per individual developer this equated to approximately 55 defect fixes per day, 6.9 fixes per hour, and a fix every 8.7 minutes. Given the rapid pace of defect resolution, the team established a peer review process as part of the overall verification and validation process. However, this subsequently became unfeasible by Day 2 of the off site. 20% of the 1,800 defects flagged were of a potentially high risk defect category, with 220 of them categorized as "likely" and user-visible, albeit with an unevaluated risk correlation. For several people in the group, it was reassuring that only 20% were high risk across all defects in the code, but it was nonetheless acknowledgment that there were real issues still in their code, and they could potentially be detected by a common user. The most common issues detected were dead code, tainted data, and forwarding null pointers. Reviewing the issues discovered by Coverity also sparked interesting discussion and education on defect patterns including questions such as 1) do we believe this defect is real? and 2) if it is a false positive, why did Coverity detect it in the first place? Of the 1,620 defects fixed, approximately 300 have gone through full validation, documentation, and testing to-date.

When we interviewed BRL-CAD's Morrison, we asked him two questions:

Q: What was most surprising throughout this process?
A: The team expected to come out with that one "aha" bug everyone knew was in the code but couldn't find. We found and fixed a lot of crash-causing and other key bugs though, but we didn't find that diamond in the rough.

Q: What was the most enlightening part of this process?
A: That even in a codebase that has been well maintained over the past 30 years, there were still a variety of latent issues detected by Coverity. Looking through the logic in Coverity reports, there were 100 different conditionals to get to one case, but looking at the report you could see how it could get to that point and cause the system to crash. It could be an obscure one-in-a-million defect, but they are critical defects and issues you can't reproduce because of the particular circumstance required to trigger it.

What's next for BRL-CAD? While the off-site was very effective in resolving most of the defects, each commit must still be properly documented and tested. Once this is complete, the intent is to make Coverity Scan a part of BRL-CAD's weekly build process and have the defect reports go out to the team with each build.

According to Morrison, "Coverity Scan is an absolutely fantastic resource for the open source community. Open source in general is a collection of random parties and individuals all with different development philosophies. Coverity Scan brings an unbiased third party evaluator to review code. With a known false positive of less than 10%, chances are when Coverity flags an issue, it means you really have these defects in your code. Most projects care about quality passionately and Coverity Scan gives you an objective view of quality."

A Comparison of Open Source and Proprietary Code

For the first time, this year's analysis included an examination of proprietary code, including a sample of over 300 million lines of code from 41 proprietary codebases of anonymous Coverity users to see what comparisons we could draw between open source and commercial projects that have adopted static analysis. These codebases represent a variety of industries and span a comparable length of adoption time as open source, from less than one year to over five years. To make the data uniform, we selected only proprietary codebases from users who provided us with a detailed breakdown that allowed us to compute defect density for high- and medium-impact defects. We used data from 43 customers, spanning multiple verticals and codebase sizes.

This year's analysis led us to our final set of findings:

> ***Proprietary codebases that leverage automated testing such as static analysis have quality above average for the software industry.***
>
> ***Open source quality is on par with proprietary code quality, particularly in cases where codebases are of similar size.***

The average codebase size for proprietary codebases in our sample is 7.5 million lines of code, significantly larger than the average for open source software included in our analysis. Therefore, to make a more direct comparison we looked at the defect density of proprietary code against open source codebases of similar size. The average defect density for proprietary codebases of Coverity users is .64, which is better than the average defect density of 1.0 for the software industry. We also found that open source code quality is on par with proprietary code quality for codebases of similar size. For instance, Linux 2.6 has nearly 7 million lines of code and a defect density of .62, which is roughly identical to that of its proprietary codebase counterparts.

We also looked at average defect density for proprietary code based upon vertical, split out by safety-critical and non-safety-critical industries. Safety-critical industries represented include medical device, automotive, power and energy, industrial automation, and aerospace and defense. Industries represented in the non-safety-critical category include financial services, software and internet, telecommunications, and diversified electronics. As would be expected, the average defect density for safety-critical industries is .32, while the average defect density for non-safety-critical industries is .68 in our analysis. This is likely due to the safety-critical industries' lower tolerance for defects, requirement of testing methods as part of regulatory verification and validation processes for safety-critical software, and length of time as a user of static analysis (safety-critical companies tend to be longer-term users of Coverity). However, in all cases, this again illustrates the direct correlation between adoption of static analysis and code quality, as well as an ongoing quality improvement as length of adoption time increases.

Conclusion and Next Steps for Coverity Scan

This year's report demonstrates a simple-yet-powerful conclusion: development projects, regardless of project type, size, or industry, that make a commitment to software quality through the adoption of development testing as a part of their development workflow, reap the benefits of high code quality and continue to see quality improvements over time.

The days of labeling open source software as something separate from and unequal to proprietary software are becoming a thing of the past. Commercial projects should continue to implement measures for better visibility into code quality across the software supply chain, but commercial projects shouldn't be afraid to integrate open source into their software supply chains based upon defect density. In reality, the lines between commercial projects and open source projects are still blurring due to the continued rise of open source software adoption in commercial software development. Our results indicate that open source projects with an active development community and commitment to quality are successfully implementing static analysis and other quality control measures in the same way as commercial projects.

A continued focus for Coverity Scan in 2012 and beyond will be providing visibility into open source projects via the Coverity Software Integrity Report as a baseline for quality. In 2012, we will make it even easier for open source projects to onboard and integrate Coverity Scan results into their existing workflows. In addition, with the formation of the Coverity Security Research Laboratory we also intend to foster collaboration between our security researchers and the open source community to further the quality and security of open source code.

Finally, we would like to thank the open source projects participating in Coverity Scan for their continued support and participation, and we look forward to extending this service to additional projects in 2012.

About Coverity

Coverity, Inc., (www.coverity.com), the development testing leader, is the trusted standard for companies that need to protect their brands and bottom lines from software failures. More than 1,100 Coverity customers use Coverity's development testing suite of products to automatically test source code for software defects that could lead to product crashes, unexpected behavior, security breaches, or catastrophic failure. Coverity is a privately held company headquartered in San Francisco. Coverity is funded by Foundation Capital and Benchmark Capital.

Appendix A: Coverity Scan 2011 Open Source Integrity Report Aggregate Findings

TABLE 1: COVERITY SCAN DATA	
Total Lines of Code Scanned	37,446,469
Average Defect Density (as of 12/31/2011)	0.45
Number of Outstanding Defects (as of 12/31/2011)	16,884
Number of Defects Fixed in 2011	6,133

TABLE 2: MOST COMMON DEFECTS OUTSTANDING		
Defect Category	**Quantity**	**Impact**
Control Flow Issues	3,128	Medium
Null Pointer Dereferences	2,818	Medium
Uninitialized Variables	2,051	High
Memory - Corruptions	1,551	High
Error Handling Issues	1,535	Medium
Resource Leaks	1,384	High
Integer Handling Issues	1,368	Medium
Memory - Illegal Access	1,136	High
Insecure Data Handling	824	Medium
Incorrect Expression	550	Medium
Concurrent Access Violations	253	Medium
API Usage Errors	178	Medium
Program Hangs	107	Medium
Class Hierarchy Inconsistencies	1	Medium

TABLE 3: MOST COMMON DEFECTS FIXED		
Defect Category	**Quantity**	**Impact**
Null Pointer Dereferences	1,299	Medium
Integer Handling Issues	881	Medium
Resource Leaks	742	High
Control Flow Issues	599	Medium
Uninitialized variables	588	High
Memory - Corruptions	482	High
Incorrect Expression	478	Medium
Error Handling Issues	476	Medium
Memory - Illegal Access	308	High
Insecure Data Handling	164	Medium
Concurrent Access Violations	51	Medium
API Usage Errors	34	Medium
Program Hangs	31	Medium
Class Hierarchy Inconsistencies	0	Medium

Appendix B: Coverity Software Integrity Report for Linux 2.6

coverity®

Software Integrity Report

Project Name: Linux-2.6

Version: 2.6

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 6,849,378

Project Defect Density: 0.62

High-Impact and Medium-Impact Defects: 4261

Target Level 1 ACHIEVED

Company Name:	Linux	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 1:31:39 PM	Coverity email:	integrityrating@coverity.com
Report ID:	25e067e9-7189-4d26-a03b-cfcf79319c82		

The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.

Software Integrity Report

Project Name: Linux-2.6

Version: 2.6

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 6,849,378

Project Defect Density: 0.62

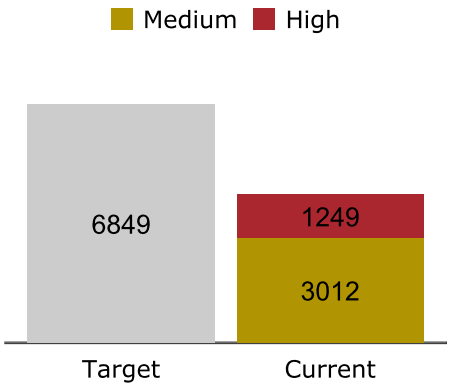
High-Impact and Medium-Impact Defects: 4261

Target Level 1 **ACHIEVED**

Company Name:	Linux	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 1:31:39 PM	Coverity email:	integrityrating@coverity.com
Report ID:	25e067e9-7189-4d26-a03b-cfcf79319c82		

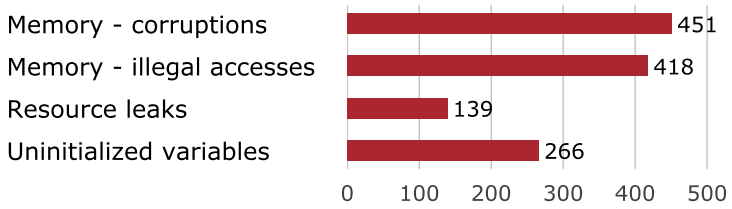
The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.



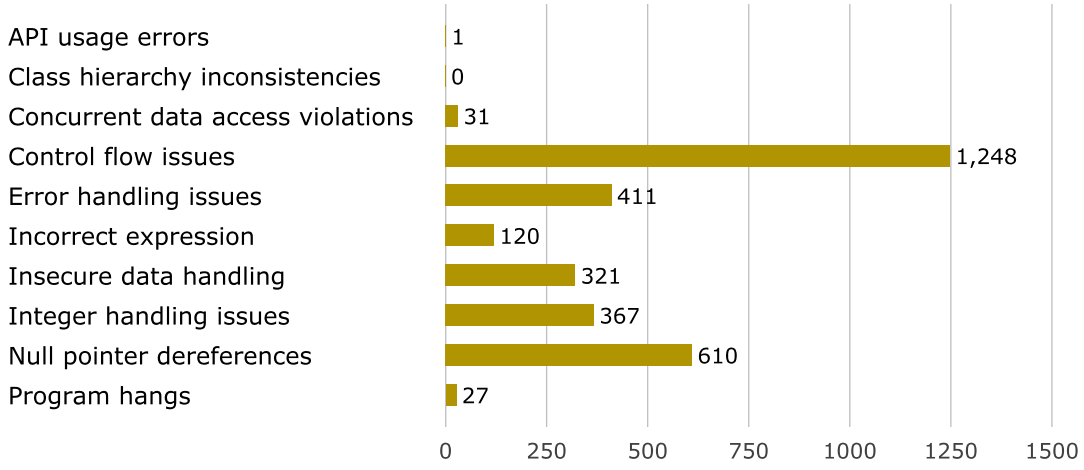
High-Risk Defects

High-impact defects that cause crashes, program instability, and performance problems.



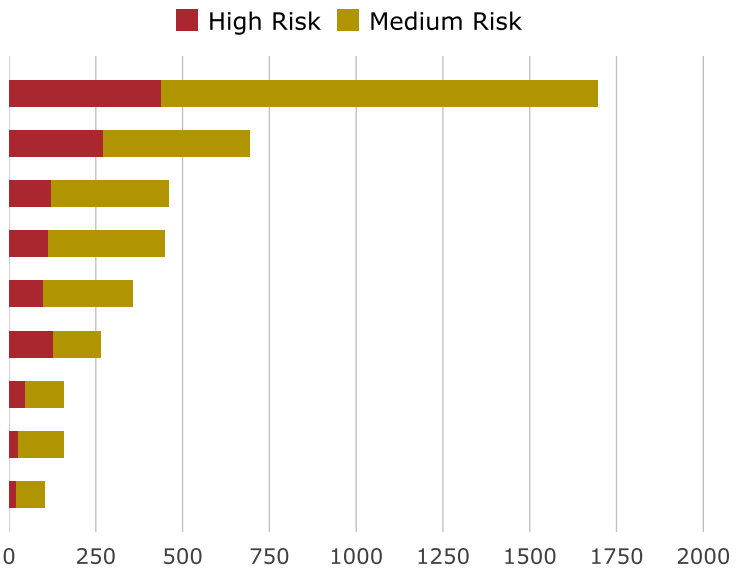
Medium-Risk Defects

Medium-impact defects that cause incorrect results, concurrency problems, and system freezes.



Defect Risk by Component

Component	Owner	Defect Density
Linux.MiscDrivers		0.61
Linux.NetDrivers		0.61
Linux.FS		0.65
Linux.StagingDri		0.97
Linux.Networking		0.67
Linux.Other		0.48
Linux.Sound		0.38
Linux.Kernel		0.95
Linux.ISDNDrive		0.73



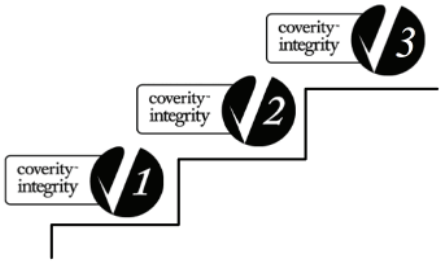
Coverity Software Integrity Report

The Coverity Software Integrity Rating is an objective standard used by developers, management, and business executives to assess the software integrity level of the code they are shipping in their products and systems.

Coverity rating requirements are based on an assessment of several factors:

- Defect density: For a given component or code base, the number of high-risk and medium-risk defects found by static analysis divided by the lines of code analyzed. Defect density excludes fixed defects and defects dismissed as false positives or intentional. For example, if there are 100 high-risk and medium-risk defects found by static analysis in a code base of 100,000 lines of code, the defect density would be 100/100,000 = 1 defect per thousand lines of code.
- Major severity defects: Developers can assess the severity of defects by marking them as Major, Moderate, or Minor (customizations might affect these labels). We consider all defects assigned a severity rating of Major to be worth reporting in the Integrity Report regardless of their risk level because the severity rating is manually assigned by a developer who has reviewed the defect.
- False positive rate: Developers can mark defect reports as false positives if they are not real defects. We consider a false positive rate of less than 20% to be normal for Coverity Static Analysis. A false positive rate above 20% indicates possible misconfiguration, incorrect inspection, use of unusual idioms in the code, or a flaw in our analysis.

Coverity Integrity Level 1 requires the software has less than or equal to 1 defect per thousand lines of code, which is approximately the average defect density for the software industry.



Coverity Integrity Level 2 requires the software has less than or equal to 0.1 defect per thousand lines of code, which is approximately at the 90th percentile for the software industry. This is a much higher bar to satisfy than Level 1. A million-line code base would have to have 100 or fewer defects to qualify for Level 2.

Coverity Integrity Level 3 This is the highest bar in the rating system today. All three of the following criteria need to be met:

- Defect density less than or equal to 0.01 defect per thousand lines of code, which is approximately in the 99th percentile for the software industry. This means that a million-line code base must have ten or fewer defects remaining. The requirement does not specify zero defects because this might force the delay of a release for a few stray static analysis defects that are not in a critical component (or else giving up on achieving a target Level 3 for the release).
- False positive rate less than 20%. If the rate is higher the results need to be audited by Coverity to qualify for this integrity rating level. A higher false positive rate indicates misconfiguration, usage of unusual idioms, or incorrect diagnosis of a large number of defects. The Coverity Static Analysis has less than 20% false positives for most code bases, so we reserve the right to audit false positives when they exceed this threshold.
- Zero defects marked as Major severity by the user. The severity of each defect can be set to Major, Moderate, or Minor. This requirement ensures that all defects marked as Major by the user are fixed because we believe that once human judgment has been applied, all Major defects must be fixed to achieve Level 3.

Level Not Achieved indicates that the target level criteria are not met. This means that the software has too many unresolved static analysis defects in it to qualify for the desired target integrity level. To achieve the target integrity level rating, more defects should be reviewed and fixed.

How to Use Your Software Integrity Rating

Set software integrity standards for your projects, products, and teams.
It is often difficult for developers and development management to objectively compare the integrity of code bases, projects, and products. The Coverity Software Integrity Rating is a way to create "apples-to-apples" comparisons and promote the success of development teams that consistently deliver highly-rated software code and products. Development teams can also use these ratings as objective evidence to satisfy requirements for quality and safety standards.

Audit your software supply chain.
It is challenging for companies to assess the integrity of software code from suppliers and partners that they integrate with their offerings. The Coverity Software Integrity Rating is a way to help companies create a common measurement of software integrity across their entire software supply chain.

Promote your commitment to software integrity.
The integrity of your software has a direct impact on the integrity of your brand. Showcasing your commitment to software integrity is a valuable way to boost your brand value. It indicates that they are committed to delivering software that is safe, secure, and performs as expected.

Appendix C: Coverity Software Integrity Report for PHP 5.3

coverity®

Software Integrity Report

Project Name: PHP

Version:

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 537,871

Project Defect Density: 0.20

High-Impact and Medium-Impact Defects: 97

Target Level 1 ACHIEVED

Company Name:	PHP	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 1:48:21 PM	Coverity email:	integrityrating@coverity.com
Report ID:	2d7d16eb-7206-4241-af9d-0c982a14ff16		

The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.

Software Integrity Report

Project Name: PHP

Version:

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 537,871

Project Defect Density: 0.20

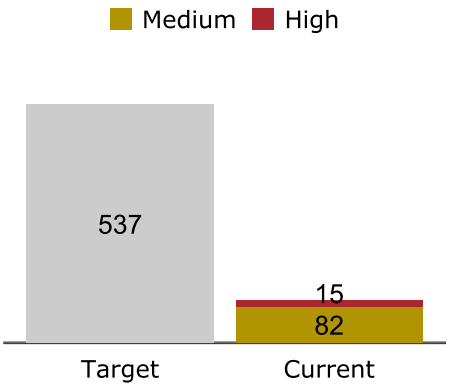
High-Impact and Medium-Impact Defects: 97

Target Level 1 **ACHIEVED**

Company Name:	PHP	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 1:48:21 PM	Coverity email:	integrityrating@coverity.com
Report ID:	2d7d16eb-7206-4241-af9d-0c982a14ff16		

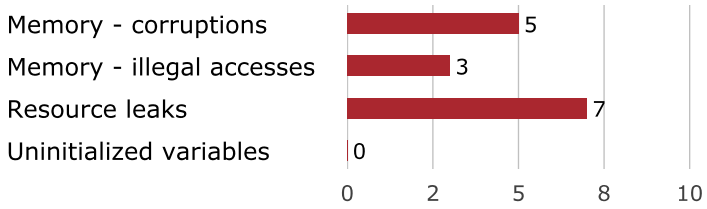
The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.



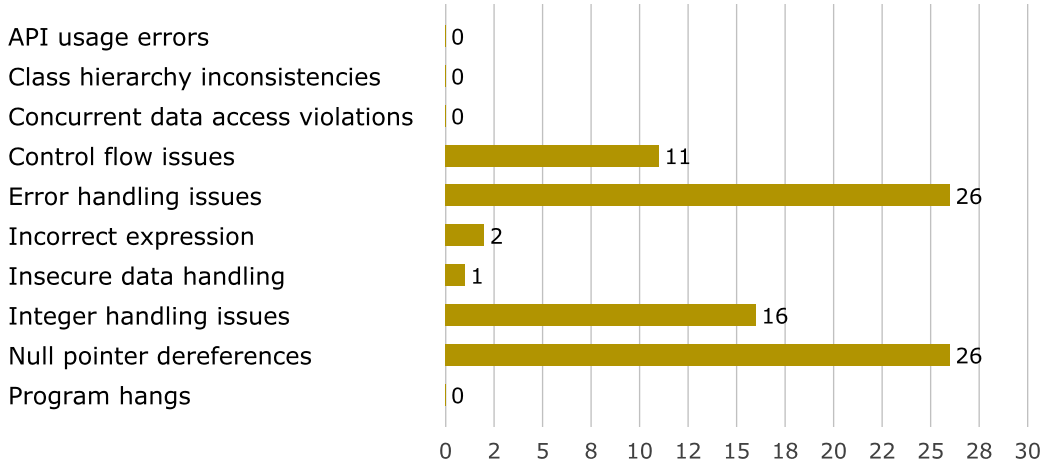
High-Risk Defects

High-impact defects that cause crashes, program instability, and performance problems.



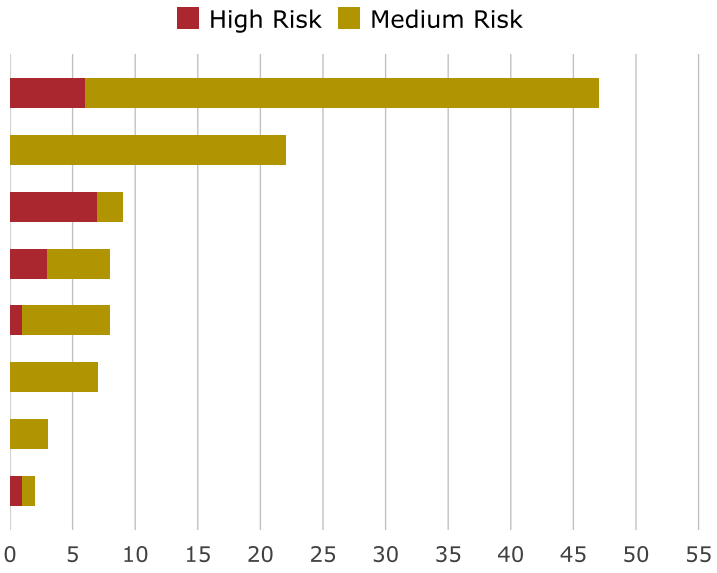
Medium-Risk Defects

Medium-impact defects that cause incorrect results, concurrency problems, and system freezes.



Defect Risk by Component

Component	Owner	Defect Density
PHP.SQLite Lib		0.61
PHP.Other		0.10
PHP.ext_date		0.17
PHP.Phar		0.55
PHP.Zend		0.09
PHP.ext_standard		0.14
PHP.Main		0.14
PHP.PDO		0.30



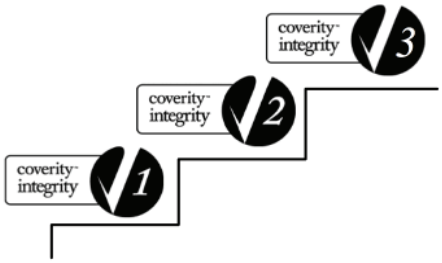
Coverity Software Integrity Report

The Coverity Software Integrity Rating is an objective standard used by developers, management, and business executives to assess the software integrity level of the code they are shipping in their products and systems.

Coverity rating requirements are based on an assessment of several factors:

- Defect density: For a given component or code base, the number of high-risk and medium-risk defects found by static analysis divided by the lines of code analyzed. Defect density excludes fixed defects and defects dismissed as false positives or intentional. For example, if there are 100 high-risk and medium-risk defects found by static analysis in a code base of 100,000 lines of code, the defect density would be 100/100,000 = 1 defect per thousand lines of code.
- Major severity defects: Developers can assess the severity of defects by marking them as Major, Moderate, or Minor (customizations might affect these labels). We consider all defects assigned a severity rating of Major to be worth reporting in the Integrity Report regardless of their risk level because the severity rating is manually assigned by a developer who has reviewed the defect.
- False positive rate: Developers can mark defect reports as false positives if they are not real defects. We consider a false positive rate of less than 20% to be normal for Coverity Static Analysis. A false positive rate above 20% indicates possible misconfiguration, incorrect inspection, use of unusual idioms in the code, or a flaw in our analysis.

Coverity Integrity Level 1 requires the software has less than or equal to 1 defect per thousand lines of code, which is approximately the average defect density for the software industry.



Coverity Integrity Level 2 requires the software has less than or equal to 0.1 defect per thousand lines of code, which is approximately at the 90th percentile for the software industry. This is a much higher bar to satisfy than Level 1. A million-line code base would have to have 100 or fewer defects to qualify for Level 2.

Coverity Integrity Level 3 This is the highest bar in the rating system today. All three of the following criteria need to be met:

- Defect density less than or equal to 0.01 defect per thousand lines of code, which is approximately in the 99th percentile for the software industry. This means that a million-line code base must have ten or fewer defects remaining. The requirement does not specify zero defects because this might force the delay of a release for a few stray static analysis defects that are not in a critical component (or else giving up on achieving a target Level 3 for the release).
- False positive rate less than 20%. If the rate is higher the results need to be audited by Coverity to qualify for this integrity rating level. A higher false positive rate indicates misconfiguration, usage of unusual idioms, or incorrect diagnosis of a large number of defects. The Coverity Static Analysis has less than 20% false positives for most code bases, so we reserve the right to audit false positives when they exceed this threshold.
- Zero defects marked as Major severity by the user. The severity of each defect can be set to Major, Moderate, or Minor. This requirement ensures that all defects marked as Major by the user are fixed because we believe that once human judgment has been applied, all Major defects must be fixed to achieve Level 3.

Level Not Achieved indicates that the target level criteria are not met. This means that the software has too many unresolved static analysis defects in it to qualify for the desired target integrity level. To achieve the target integrity level rating, more defects should be reviewed and fixed.

How to Use Your Software Integrity Rating

Set software integrity standards for your projects, products, and teams.
It is often difficult for developers and development management to objectively compare the integrity of code bases, projects, and products. The Coverity Software Integrity Rating is a way to create "apples-to-apples" comparisons and promote the success of development teams that consistently deliver highly-rated software code and products. Development teams can also use these ratings as objective evidence to satisfy requirements for quality and safety standards.

Audit your software supply chain.
It is challenging for companies to assess the integrity of software code from suppliers and partners that they integrate with their offerings. The Coverity Software Integrity Rating is a way to help companies create a common measurement of software integrity across their entire software supply chain.

Promote your commitment to software integrity.
The integrity of your software has a direct impact on the integrity of your brand. Showcasing your commitment to software integrity is a valuable way to boost your brand value. It indicates that they are committed to delivering software that is safe, secure, and performs as expected.

Appendix D: Coverity Software Integrity Report for PostgreSQL 9.1

coverity®

Software Integrity Report

Project Name: PostgreSQL

Version:

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 1,105,634

Project Defect Density: 0.21

High-Impact and Medium-Impact Defects: 233

Target Level 1 ACHIEVED

Company Name:	PostgreSQL	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 2:59:57 PM	Coverity email:	integrityrating@coverity.com
Report ID:	bd3d4b44-04db-4a12-ae73-a8380c19e5c8		

The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.

coverity®

Software Integrity Report

Project Name: PostgreSQL

Version:

Project Description: 2011 Open Source Scan

Project Details:

Lines of Code Inspected: 1,105,634

Project Defect Density: 0.21

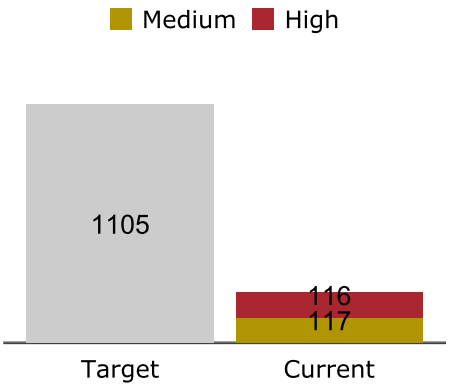
High-Impact and Medium-Impact Defects: 233

Target Level 1 **ACHIEVED**

Company Name:	PostgreSQL	Coverity Product:	Static Analysis
Point of Contact:	Scan	Product Version:	5.2.1
Client email:	scan-admin@coverity.com	Coverity Point of Contact:	Integrity Report
Report Date:	Feb 6, 2012 2:59:57 PM	Coverity email:	integrityrating@coverity.com
Report ID:	bd3d4b44-04db-4a12-ae73-a8380c19e5c8		

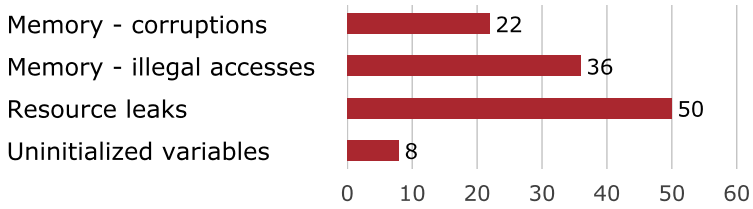
The Coverity Integrity Rating Program provides a standard way to objectively measure the integrity of your own software as well as software you integrate from suppliers and the open source community. Coverity Integrity Ratings are established based on the number of defects found by Coverity® Static Analysis when properly configured, as well as the potential impact of defects found. Coverity Integrity Ratings are indicators of software integrity, but do not guarantee that certain kinds of defects do not exist in rated software releases or that a release is free of defects. Coverity Integrity Ratings do not evaluate any aspect of the software development process used to create the software.

A Coverity customer interested in certifying their ratings can submit this report and the associated XML file to integrityrating@coverity.com. All report data will be assessed and if the Coverity Integrity Rating Program Requirements are met, Coverity will certify the integrity level achieved for that code base, project, or product.



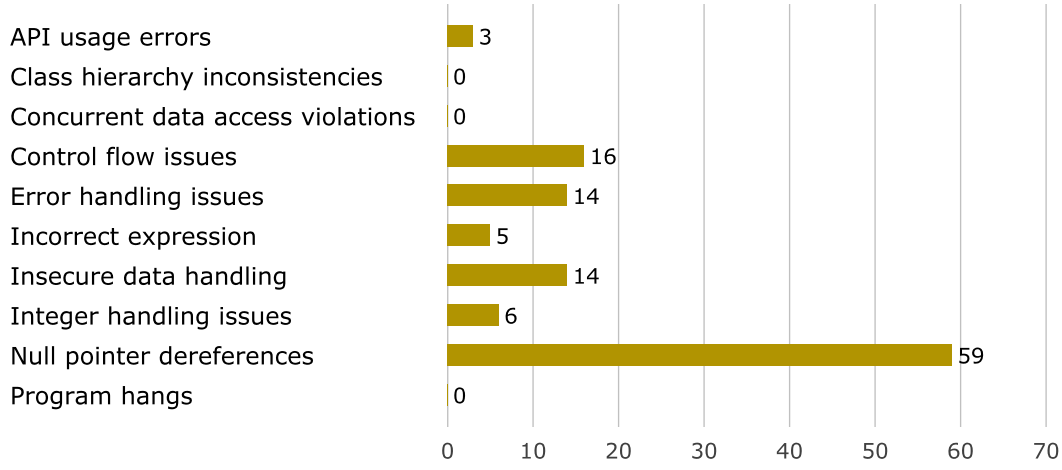
High-Risk Defects

High-impact defects that cause crashes, program instability, and performance problems.



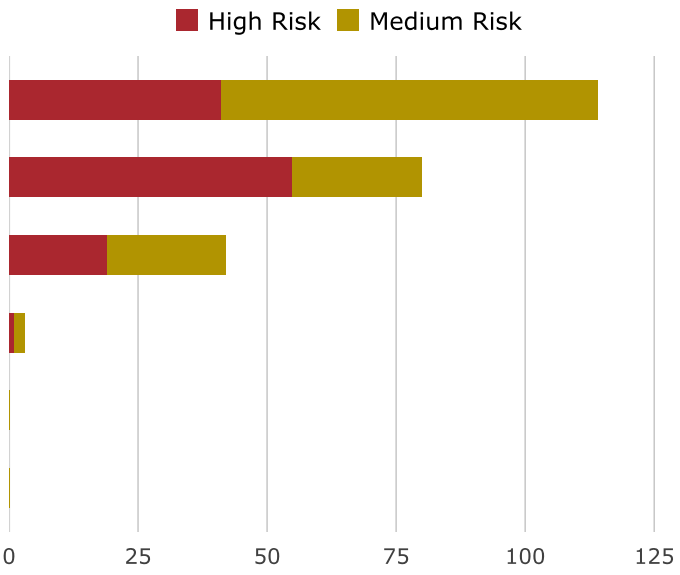
Medium-Risk Defects

Medium-impact defects that cause incorrect results, concurrency problems, and system freezes.



Defect Risk by Component

Component	Owner	Defect Density
PG.Backend		0.13
PG.pgsql		0.65
PG.Interfaces		0.45
PG.Port		1.15
PG.Other		0.00
PG.usr		0.00



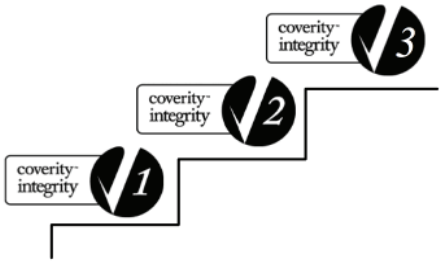
Coverity Software Integrity Report

The Coverity Software Integrity Rating is an objective standard used by developers, management, and business executives to assess the software integrity level of the code they are shipping in their products and systems.

Coverity rating requirements are based on an assessment of several factors:

- Defect density: For a given component or code base, the number of high-risk and medium-risk defects found by static analysis divided by the lines of code analyzed. Defect density excludes fixed defects and defects dismissed as false positives or intentional. For example, if there are 100 high-risk and medium-risk defects found by static analysis in a code base of 100,000 lines of code, the defect density would be 100/100,000 = 1 defect per thousand lines of code.
- Major severity defects: Developers can assess the severity of defects by marking them as Major, Moderate, or Minor (customizations might affect these labels). We consider all defects assigned a severity rating of Major to be worth reporting in the Integrity Report regardless of their risk level because the severity rating is manually assigned by a developer who has reviewed the defect.
- False positive rate: Developers can mark defect reports as false positives if they are not real defects. We consider a false positive rate of less than 20% to be normal for Coverity Static Analysis. A false positive rate above 20% indicates possible misconfiguration, incorrect inspection, use of unusual idioms in the code, or a flaw in our analysis.

Coverity Integrity Level 1 requires the software has less than or equal to 1 defect per thousand lines of code, which is approximately the average defect density for the software industry.



Coverity Integrity Level 2 requires the software has less than or equal to 0.1 defect per thousand lines of code, which is approximately at the 90th percentile for the software industry. This is a much higher bar to satisfy than Level 1. A million-line code base would have to have 100 or fewer defects to qualify for Level 2.

Coverity Integrity Level 3 This is the highest bar in the rating system today. All three of the following criteria need to be met:

- Defect density less than or equal to 0.01 defect per thousand lines of code, which is approximately in the 99th percentile for the software industry. This means that a million-line code base must have ten or fewer defects remaining. The requirement does not specify zero defects because this might force the delay of a release for a few stray static analysis defects that are not in a critical component (or else giving up on achieving a target Level 3 for the release).
- False positive rate less than 20%. If the rate is higher the results need to be audited by Coverity to qualify for this integrity rating level. A higher false positive rate indicates misconfiguration, usage of unusual idioms, or incorrect diagnosis of a large number of defects. The Coverity Static Analysis has less than 20% false positives for most code bases, so we reserve the right to audit false positives when they exceed this threshold.
- Zero defects marked as Major severity by the user. The severity of each defect can be set to Major, Moderate, or Minor. This requirement ensures that all defects marked as Major by the user are fixed because we believe that once human judgment has been applied, all Major defects must be fixed to achieve Level 3.

Level Not Achieved indicates that the target level criteria are not met. This means that the software has too many unresolved static analysis defects in it to qualify for the desired target integrity level. To achieve the target integrity level rating, more defects should be reviewed and fixed.

How to Use Your Software Integrity Rating

Set software integrity standards for your projects, products, and teams.
It is often difficult for developers and development management to objectively compare the integrity of code bases, projects, and products. The Coverity Software Integrity Rating is a way to create "apples-to-apples" comparisons and promote the success of development teams that consistently deliver highly-rated software code and products. Development teams can also use these ratings as objective evidence to satisfy requirements for quality and safety standards.

Audit your software supply chain.
It is challenging for companies to assess the integrity of software code from suppliers and partners that they integrate with their offerings. The Coverity Software Integrity Rating is a way to help companies create a common measurement of software integrity across their entire software supply chain.

Promote your commitment to software integrity.
The integrity of your software has a direct impact on the integrity of your brand. Showcasing your commitment to software integrity is a valuable way to boost your brand value. It indicates that they are committed to delivering software that is safe, secure, and performs as expected.