



coverity

Open Source SCAN project

- Launched, March 2006
- DHS sponsored “Open Source Hardening Project”
 - 2006-2009
- Using Coverity’s commercial static analysis product to identify bugs at the source code level
- 35 open source projects on day one
- Since grown to 250+ projects
- Over 11,000 bugs fixed

Scan over time



Identifying Bugs

scan.coverity.com brought to you by **coverity**

"Coverity's static source code analysis has proven to be an effective step towards furthering the quality and security of Linux."

- Andrew Morton, lead kernel maintainer

Next Up:

- Open Source Automated Quality Symposium:** Participate in the evolution of scan.coverity.com and meet with peers in the open source community who are leveraging our new technology. Dates and details to come...
- We will also be putting up an online Forum, FAQ, and additional resources so stay tuned!**

Download the full report:
["Measuring software quality: A study of open source software"](#)

Free Trial
 How Many Defects are in Your Code?

How does your code compare to the LAMP stack? Coverity offers a free trial on your code with the same

Automating and Accelerating Open Source Quality

In collaboration with Stanford University, Coverity is establishing a new baseline for software quality and security in open source based on the analysis of over 30 of the most critical and widely used open source projects in the world. Under a contract with the Department of Homeland Security, we apply the latest innovation in automated defect detection to uncover some of the most critical types of bugs found in software.

"Coverity has found bugs in parts of Samba that we had previously considered robust and tested. It's like having a developer on the team with an inhuman attention to detail...It's making a major contribution to the code quality of the Samba project."
 -Jeremy Allison, Samba Team

We are making the results of our automated analysis available to the maintainers within the open source community. Additional projects will be added over time. Please click on the registration link to gain access. We have been receiving a high number of requests so we'll do our best to respond within 24 hours.

Project	Current # Defects	Original # Defects	Lines of Code	Defects / KLOC	View Results	Please Register to View Results
AMANDA	0	108	88,879	0.000	Sign In	Register
apache-htpd	26	92	127,742	0.204	Sign In	Register
ethereal	10	143	1,131,049	0.009	Sign In	Register
Firebird	197	163	251,334	0.784	Sign In	Register
Firefox	50	108	305,380	0.164	Sign In	Register
FreeBSD	672	635	5,292,166	0.129	Sign In	Register
Daem	91	113	925,719	0.157	Sign In	Register
gcc	58	140	623,258	0.155	Sign In	Register

SCAN COVERITY.COM brought to you by **coverity**

MAIN: ABOUT SCAN, FAQ, DEVELOPER FAQ | SCAN LEADER: FAQ, RUND 1 - 53 PROJECTS, RUND 0 - 100 PROJECTS, ALL PROJECTS | AMANDA CHART, SAMBA CHART | POLICY STATEMENT

COVERITY'S STATIC SOURCE CODE ANALYSIS HAS PROVEN TO BE AN EFFECTIVE STEP TOWARDS FURTHERING THE QUALITY AND SECURITY OF LINUX.

ACCELERATING OPEN SOURCE QUALITY

In collaboration with Stanford University, Coverity is establishing a new baseline for software quality and security in open source based on the analysis of over 30 of the most critical and widely used open source projects in the world. Under a contract with the Department of Homeland Security, we apply the latest innovations in automated defect detection to uncover some of the most critical types of bugs found in software.

TOTAL NUMBER OF DEFECTS FIXED (SINCE 03/09/2006): 6,935

Amanda's developers fix over 40% of the Scan's detected defects with a single reading of the Scan analysis for that issue. The red defects were MEMORY LEAKS. In the coming weeks and months, we'll be providing additional information on this Amanda graph, and graphs for other projects. We will also provide more detailed examinations of the progress projects have made, and how the analysis is being used.

Other charts are available. See the list under Progress Charts in the navbar.

OPEN SOURCE PROJECT LEADER BOARD

Rank	Project	Defect Summary	Lines of Code	Defects / KLOC	View Results
1	AMANDA	Fixed: 0, Unreported: 0	95,086	0.000	Sign In
1	AMANDA	Fixed: 0, Unreported: 0	294,539	0.000	Sign In
1	ntp	Fixed: 0, Unreported: 0	130,789	0.000	Sign In
1	DynanDAR	Fixed: 0, Unreported: 0	68,279	0.000	Sign In

NEWS

Happy First Birthday, Scan

Coverity Names David Muesel as Open Source Strategist

Coverity detects a **gapsite_bug** in Windows that allows any user with a high to gain root privileges

Amanda releases major version (2.5) of the popular backup and recovery software with milestones of **Coverity defect**

Scan.coverity.com results in **over 1,000 patches** to projects in the Free-software

MEMBERSHIP

Coverity Study Partner **JAMP Code Quality**

WEEK
 Critical Code Near-Source Security Project

Architecture Library

scan.coverity.com brought to you by **coverity**

Architecture Library

COVERITY'S STATIC SOURCE CODE ANALYSIS HAS PROVEN TO BE AN EFFECTIVE STEP TOWARDS FURTHERING THE QUALITY AND SECURITY OF LINUX.

Fig 11: 2006 - Measurement of the Coverity Architecture Library

Architecture measurement data of over 2,000 Open Source Projects from available at Coverity Scan Library

Publicly Available Library to Help Community Determine Use of Open Source and Learn Successful Substitutes

Scan the source code of your Open Source projects to determine if they are using any of the libraries in the Architecture Library. The Architecture Library is a collection of open source projects that have been analyzed by Coverity. The results of the analysis are available in the Architecture Library. The Architecture Library is a collection of open source projects that have been analyzed by Coverity. The results of the analysis are available in the Architecture Library.

MEMBERSHIP

Coverity Study Partner **JAMP Code Quality**

WEEK
 Critical Code Near-Source Security Project

Supporting more open source projects

Lessons Learned from Evaluating Software



- Communications of the ACM

A Few Billion Lines of Code Later:

Code Later:

Using Static Analysis to Find Bugs in the Real World

(including open source)

The screenshot shows the Communications of the ACM website. The main article is titled "A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World" by W. Bradford Paley. The article discusses the development and commercialization of a static bug-finding tool, CodeProfiles, and its application in finding generic errors such as memory corruption and data races. The article is part of the "Magazine Archive" for 2010, No. 2. The website also features a search bar, navigation links, and a sidebar with related news and resources.

Open Source Reports



- Whitepaper series - <http://scan.coverity.com/report/>
 - Open Source Report 2008
 - Open Source Report 2009

Most Commonly Found Defects

Looking at the most commonly found defects in code can help formulate ideas about what kind of code constructs may cause developers to make more errors. A type of defect might be more frequently found because it involves code constructs that are harder to understand, more frequently used, or involve a hard-to-use programming interface. Programming defects at this level are often the root cause of crashes, security vulnerabilities, and other program misbehavior.

Ranking Defect Types

From the inception of the Scan site in 2006 until May 2008, Scan discovered an aggregate of 27,752 defects among all the open source projects participating in Scan. That number increased to a total of 38,453 defects found by August 2009. In this section, we consolidated all defects across all participating open source projects and categorized them by defect types. The results are shown in the following table.

Defect Type	2008 Frequency	2009 Frequency	% Difference	Ranking Change
NULL Pointer Dereference	27.95%	27.81%	0.14% ↓	0
Resource Leak	25.73%	23.34%	2.39% ↓	0
Unintentional Ignored Expressions	9.76%	9.71%	0.05% ↓	0
Use Before Test (NULL)	8.09%	8.35%	0.25% ↑	-1
Use After Free	6.46%	5.91%	0.34% ↓	-1
Buffer Overflow (statically allocated)	6.14%	5.79%	0.55% ↓	-1
Unsafe Use of Returned NULL	5.85%	5.30%	0.55% ↓	-1
Uninitialized Values Read	5.50%	8.41%	2.91% ↑	+4
Unsafe Use of Returned Negative	3.72%	3.90%	0.18% ↑	0
Type and Allocation Size Mismatch	0.62%	1.10%	0.48% ↑	0
Buffer Overflow (dynamically allocated)	0.31%	0.21%	0.10% ↓	0
Use Before Test (negative)	0.21%	0.18%	0.03% ↓	0

Footnote: For more descriptive information on these defect types, see Appendix D.

The 2008 Scan results in the above table are generated using the 2006 version of Coverity Static Analysis. This 2006 version was also used for the 2009 analysis of open source projects on Rung 1. However, for projects on Rung 2, a newer version of Coverity Static Analysis was used to generate the 2009 results. This accounts for the two largest changes in the distribution of results from 2008 to 2009. Projects on Rung 2 had resolved all of their earlier identified resource leaks, so the latest runs have a smaller percentage of that type of defect. Improvements to the newer version of Coverity Static Analysis used for Rung 2 projects identify a large number of additional cases of uninitialized values, leading to the increase in that category between 2008 and 2009. Viewed alone, that one capability change raised uninitialized value errors to over 21% of the defect distribution for projects now on Rung 2.

Frequency of Functions by Length

The graph is quite smooth until roughly the 200 LOC mark. At that point, it becomes rougher, and the data points become more sparse. This is an indication of where our source code data set begins to become more sparse. As the Scan data set continues to grow, we would expect to see a very gradual extension of the smooth area.

While the 2008 report showed numbers that were aggregated on a per-project basis, and happened to find that averages across all projects were in the neighborhood of a modern programmer's screen size, a close-up look at the function distribution shows no discontinuity near the screen-length functions. If programmers did have any significant tendency to break functions up, by refactoring, when the screen-length boundary is passed, then the graph should show one or more sudden drops at the common screen lengths as the functions over that size are broken up, and contribute to the distribution of shorter functions to the left of the limit.

Since the graph is clearly smooth, with no such drops, programmers either have no such tendency to break functions at screen length, or do so rarely enough for it to have no viable impact.

- Coverity's Customers use large quantities of open source in their development efforts
- Every defect fixed by an open source project doesn't have to be fixed by dozens of customers

“Don't repeat yourself”

“Don't repeat yourself”

“Don't repeat someone else, either”

- Success rate of projects varies widely
- Common Best Practices have been identified, and will be incorporated into the Scan infrastructure
- Open Source is more widely used than anyone expects

- Questions?