# Coverity® Scan Open Source Report 2014

# Table of Contents

# Executive Summary

We're excited to share our annual Coverity® Scan Report analyzing the software integrity of open source projects. Since our first publication in 2009, the Coverity Scan Report has become a widely accepted standard for measuring the state of open source quality. Our 2014 report covers the largest annual sample size we've had to date: 10 billion lines of open source software code from more than 2,500 open source C/C++, Java and C# projects as well as an anonymous sample of commercial projects.

Since 2006, the Coverity Scan service has helped developers find and fix more than 240,000 defects. In 2014 alone, developers fixed nearly 152,000 defects—more than the total number of defects found from 2006-2013.

As a result of this exponential growth, the 2014 report has a wealth of new and unique detail. Looking closely at the aggregate comparison of commercial vs. open source projects, we find what we'd expect to find. As in past years, open source projects are doing an increasingly good job of addressing defects. At the same time, enterprises are doing an increasingly good job of complying with external software security standards like the Open Web Application Security Project (OWASP) Top 10 and CWE-25. They're both getting better, but in different ways.

# Coverity Scan: Through the Years

Initiated in 2006 with the U.S. Department of Homeland Security, the Coverity® Scan service began as a joint public-private sector research project, focused on improving open source software quality and security. Synopsys now manages the Coverity Scan Service project, providing our development testing technology as a free service to the open source community to help developers build quality and security into their software development process. The Coverity Scan service enables open source developers to scan—or test—their C#, Java, C and C++ code as they write it, flag critical quality and security defects that are difficult (if not impossible) to identify with other methods and manual reviews, and provide developers with actionable information to help them to quickly and efficiently fix the defects.

"With Coverity® Scan, we're finding new issues when they get introduced so we can jump on them faster than before."

– Dave Jones,
Linux Kernel Developer

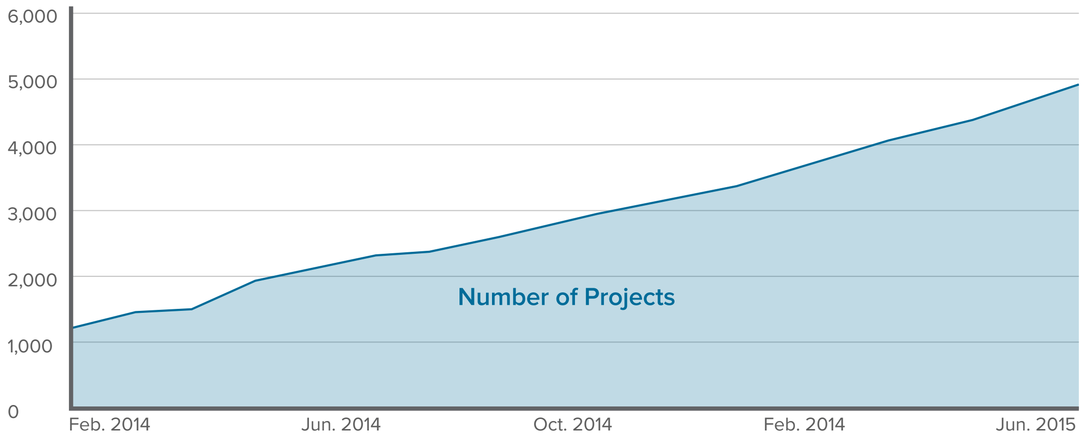


*Table 1: Total number of projects in Coverity Scan service through March 2015*

**The Coverity Scan service experienced explosive growth and by July 2015 has more than 5,100 projects.**

This growth reaffirms the power and importance of development testing and static analysis as a foundational technology for open source projects to assure the quality and security of their code.

## Coverity Scan 7.6

We recently upgraded Coverity Scan to our latest Coverity Enterprise Code Advisor 7.6 version. Key new capabilities include:

- A new checker that finds instances of vulnerable cryptographic algorithms
- A new checker that identifies "useless" function calls, so called because their return values are ignored and they have no other discernible effects
- A new Java web application security checker that finds code that stores, transmits, or logs sensitive data without protecting it adequately
- Another new Java web application security checker that finds code that applies a cryptographic hash function to password data in a cryptographically weak manner
- Several enhancements and new defect patterns for existing checkers

We have upgraded the Scan infrastructure and added new machines that allow open source users to submit builds more frequently than before. We also added an email subscription option for new defects found in selected modules, as requested by several users.

"Coverity Scan identified a buffer overflow vulnerability, CVE-2014-0064, in the path_in() function. It affects all supported versions back to 8.4."

– **PostgreSQL**

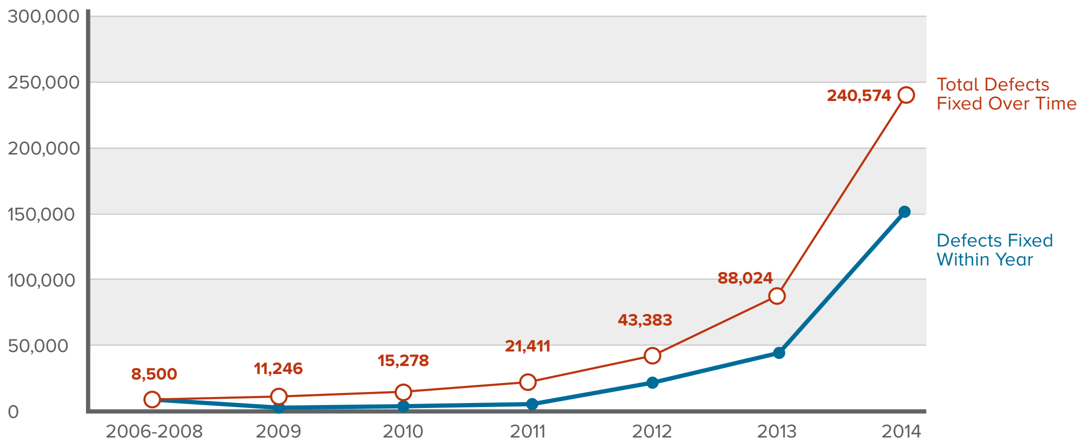


*Table 2: Defects fixed over time*

**Developers fixed almost 152,000 defects in 2014 alone—more than the number of defects found in the service from 2006-2013.**

The Coverity Scan service has long recommended a "no new defects" policy as a best practice for driving adoption of development testing. For many, seeing the vast quantities of defects the Coverity Enterprise Code Advisor analysis discovers the first time they run it is overwhelming. When faced with thousands of defects, it's difficult to know where to start. A "no new defects policy" solves that problem. It is far easier to address new defects, because the source code is still fresh in the mind of the developer and therefore will take less time to fix. In addition to ensuring they introduce no new defects, teams then also implement a plan for working on the backlog of issues, starting with the most critical.

For this 2014 report, we started with 14,000 commercial and 5,100 open source C/C++, Java and C# projects. Out of all these projects, we choose 10,000 most active projects and analyzed more than 10 billion lines of code across C/C++, Java and C# projects.

# State of Software Security

The sample open source projects in this report range in size from fewer than 5,000 lines of code (LoC) to more than 12 million (for FreeBSD). The next three largest projects in the Coverity Scan service are Linux, NetBSD and LibreOffice, with 10, 9 and 7 million LoC respectively.

## Key Findings

### 1. Open source software has a considerably lower defect density than commercial software.

If we look at cumulative defect density (number of defects per 1,000 lines of code) across all languages (C/C++, Java, C#) through 2014, then the state of open source software (defect density of 0.61), with respect to static analysis findings, is considerably better than commercial software (defect density of 0.76).

SUPPORTING DATA

| | Total LoC (in millions) | Total Projects | Defect Density Overall* |
|---|---|---|---|
| **Open Source** | 500 | 2,650 | 0.61 |
| **Commercial** | 9,100 | 8,776 | 0.76 |

### 2. The defect densities of both open source software and commercial software have continued to improve year over year.

We see this improvement when we compare the overall defect density numbers between last year and this year across C/C++, Java and C#.

SUPPORTING DATA

| | Open Source Now | Commercial Now | Open Source Last Year | Commercial Last Year |
|---|---|---|---|---|
| **LoC Scanned (millions)** | 500 | 9,100 | 260 | 700 |
| **Avgerage Defect Density** | 0.61 | 0.76 | 0.66 | 0.77 |

### 3. Commercial enterprise web application software is significantly more compliant with OWASP Top 10 than open source software.

While our Java projects are a combination of web applications and non-web application software, and OWASP is more Java web application-security focused, we believe that the OWASP Top 10 is a comprehensive compliance standard to gauge the overall security state of all Java projects.

#### TERMINOLOGY USED

**Defect density:** *Number of defects per 1000 lines of code.*

**LoC:** *Line of code.*

**OWASP:** *The Open Web Application Security Project (OWASP) is an online community dedicated to web application security. The OWASP Top 10 is a list of the most critical web application security vulnerabilities as identified by the project.*

**CWE:** *Common Weakness Enumeration (CWE) is a formal list of software weakness. CWE Top 25 is a list of the most widespread and critical errors that can lead to serious vulnerabilities in software.*

** With 98% confidence level, the variation in average defect density across all projects in our sample is +/- 0.09*

Looking at our Java defect density data through the lens of OWASP Top 10, we observe that commercial software is significantly more secure than open source software. It is important to note that even though both the commercial projects and the open source projects had the same average time of 6 months of being able to fix issues, we have observed the trend that commercial software is tackling these security vulnerabilities at a relatively faster pace than compared to open source software, which might indicate commercial software projects are driven by compliance and policy to resolve defects in this category.

SUPPORTING DATA

| | Total LoC for Projects with OWASP Top 10 Defects (in millions) | Number of Projects with OWASP Top 10 Defects | OWASP Top 10 Defects per 100K LoC[†] |
|---|---|---|---|
| **Open Source** | 14 | 84 | 8.61 |
| **Commercial** | 52 | 86 | 0.56 |

“Coverity Scan helps us find defects in our software, which—after ten years of development—are of course still to be found.”

– Mangos zero

## Key Insights

### 1. Open source and commercial software are continually improving, but in different ways.

If we look at the static analysis defect density data from this report, what we generally see is that both open source and commercial software are getting better all the time. It’s also clear that open source and commercial software are advancing in different ways.

- Open source software is becoming more feature-rich, getting better compared to previous versions of itself. What drives development work for open source projects is people needing the software to do certain things. Therefore, adding features takes precedence over bug fixing.
- Commercial software is becoming more stable and secure based on compliance standards. Commercial development is driven by competition and compliance to industry standards, which puts a higher priority on stability, security and bug fixing.

### 2. A combination of tools and methodologies will yield the best results.

If we look at recent security issues in commercial software, most of these development companies have compliance teams that enforce standards, but we still see serious vulnerabilities on an almost weekly basis. The industry needs to balance security with development speed, especially as more software projects move to agile methodology and time to market becomes more important than ever before. We believe a healthy combination of software tools, compliance standards and adherence to software development lifecycle principles is the best way forward to improve the security and quality of all software. Uniquely in our space, Synopsys is working on all three fronts to provide developers with the ultimate assurance that their code is the most secure.

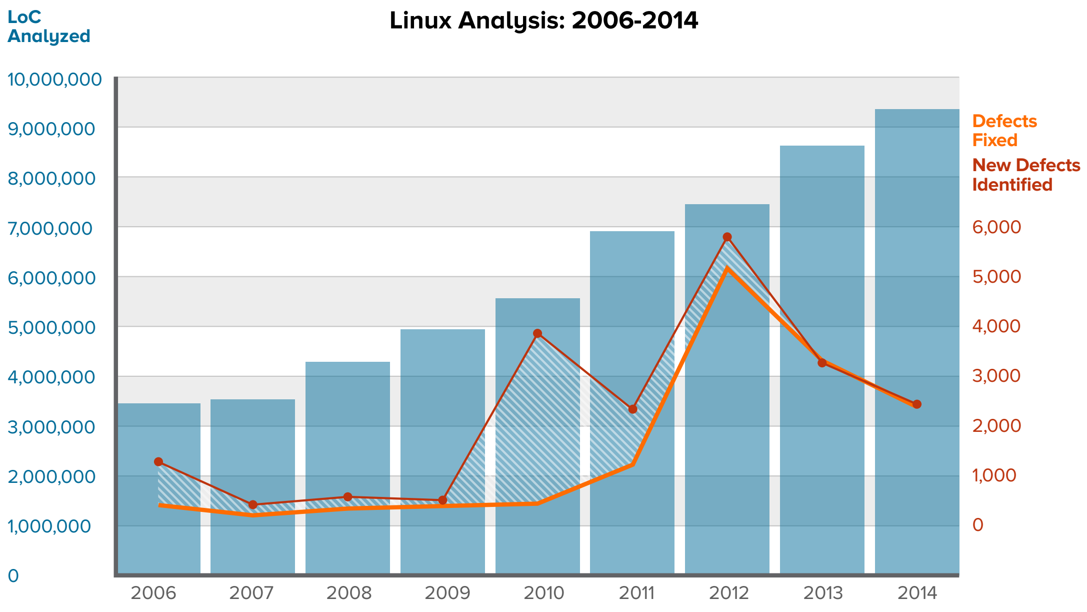
*[†] With 98% confidence level, the variation in OWASP Top 10 defects per 100K LoC across all projects in our sample is +/- 0.08*

# Linux: A Benchmark for Quality

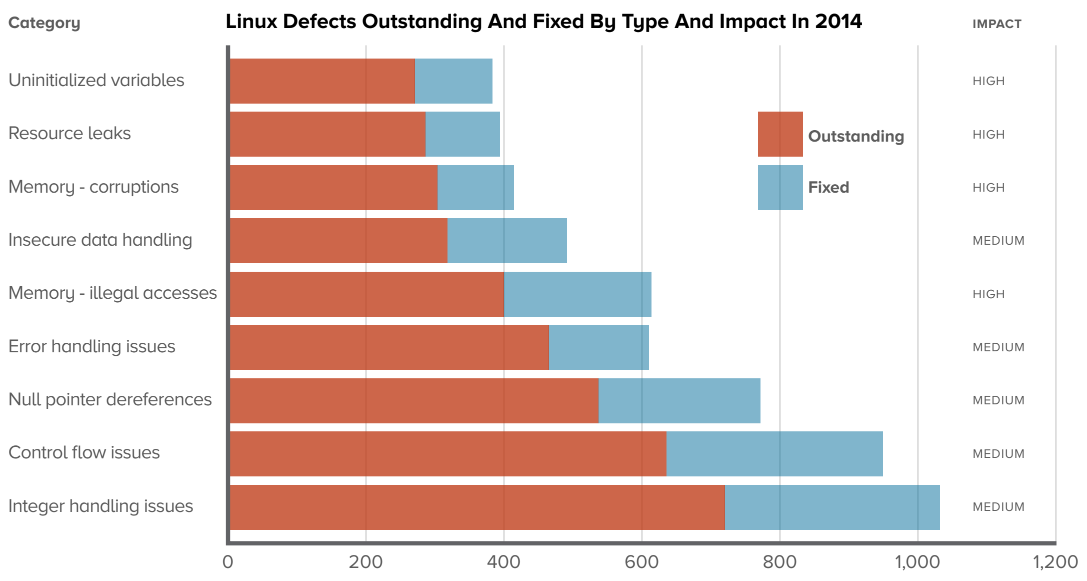
The Synopsys Coverity Scan service and Linux have a long-standing partnership on code quality that started in the early 2000s when the Coverity Scan service was still a research project in the Computer Science Laboratory at Stanford University. Since that time, both the Coverity Scan service (acquired by Synopsys in 2014) and Linux have experienced tremendous growth. One thing that has remained constant for both parties is our commitment to quality.

“As a large project with a lot of legacy code, Coverity Scan has helped us understand the quality of our code. And of course it helps us keep quality high for the better-maintained parts.”

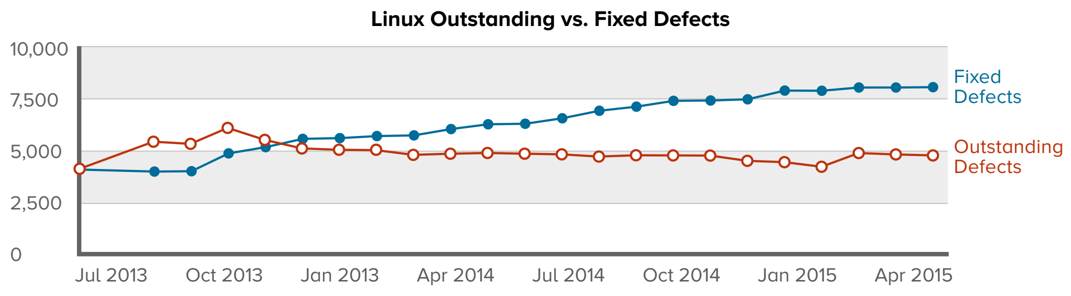
– QEMU



To help ensure highly accurate static analysis results in the Scan service, the Linux team leverages Coverity Scan modeling capabilities to help the analysis algorithms better understand the patterns and behavior of the Linux code. The analysis automatically builds models based on the source code, but it can’t always correctly infer what happens—perhaps there is no source code, like in the case of a dynamic library, or there are external effects that cannot be predicted, such as a remote procedure call.



When Dave Jones, Coverity Scan Administrator for Linux, first became involved in the Coverity Scan service in August of 2013, he established many new, smaller components for Linux to simplify management. One of the other key changes Dave implemented is a focus on newly detected defects. He believes that new defects are easier to resolve because the code is still fresh in the mind of the developer, and legacy defects are often eliminated as new code overwrites the old. We can see the progress in the graph below.



# Interview: Tim Hudson, OpenSSL

The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured and open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1.0/v1.1/v1.2) protocols as well as a full-strength general purpose cryptography library. A worldwide community of volunteers manages the project, using the Internet to communicate, plan and develop the OpenSSL toolkit and its related documentation.

The OpenSSL Project started actively using Coverity® Scan as part of its integrated development process after the Heartbleed bug. We invited Tim Hudson, co-founder of the OpenSSL Project, to share his thoughts about Coverity Scan with us for this report.

## 1. What has your overall experience with the Coverity Scan service been like?

Coverity Scan has been a useful service for the project in terms of providing access to an effective commercial static code analyzer. We have found the information from the reports useful in guiding specific review activities. OpenSSL is certainly a challenging code base for static code analyzers, but where we have encountered false positives, the Coverity Scan support team has been quick to respond. That level of vendor support makes a substantial difference when working with code analyzers.

## 2. As a participant in the Coverity Scan service, what have been the greatest benefits the OpenSSL project has received?

Fast responses when the Coverity Scan tools are updated to catch newly discovered issues to highlight potential areas where similar code may be found. This aids in dealing with the complexity of a security toolkit and variety of different interfaces provided. Basically, if a new issue is discovered, the Coverity Scan tool helps find other instances of a similar nature.

## 3. Can you share some insight on how you tackled the Heartbleed defect internally?

Once it was clear there was an issue that needed resolution, the team worked to respond with updates to fix the specific reported issues. We also used the Coverity Scan report once the tools were updated to look for other potential areas where that particular bug may have occurred. That helped focus the review activities for the team members. A substantial amount of the effort involved in a security fix release is the coordination with the downstream vendors who sit between the project team and the ultimate end users.

## 4. How has discovery and media attention for the Heartbleed bug changed the development practice for OpenSSL project?

The project has seen some substantial changes in the last year since Heartbleed. The subsequent publicity increased the level of awareness of how under-resourced the OpenSSL Project was. With a range of sponsor donations and the support from the Linux Foundation Core Infrastructure Initiative (http://www.linuxfoundation.org/programs/core-infrastructure-initiative) we now have four team members working full time on OpenSSL. We also went on an active recruitment drive, and as a result, have a much larger set of volunteers than previously. This has enabled us to start tackling more of the backlog of issues and address many of the long-term objectives. We now have published a roadmap for the project and are steadily working through the objectives within the roadmap.

"Within minutes, we were able to narrow down and fix some significant resource leaks that we were totally unaware even existed. We use Coverity at work, and now we can use it at home as well!"

**– Crane, C# open source project**

## 5. What are some of the unique challenges with writing and maintaining secure crypto code (as opposed to non-crypto code)?

A defect in an application is simply unfortunate; a defect in a security toolkit can have catastrophic consequences. This necessitates a much more cautious approach to additions to the code base. Security trumps features.

## 6. Do you have a dedicated team that is designed to ensure security?

The team participates in active reviews of code prior to release—each change to the main source repository is reviewed prior to being committed. As we maintain multiple branches of the code base, the pending changes for the next major release are up and visible for community feedback. We do get a lot of feedback—much of it conflicting—but that is part of being an open source project. Anyone is free to offer a viewpoint for consideration. We have also actively encouraged the independent audit of the OpenSSL code base that is now under way (sponsored by the Linux Foundation).

## 7. Do you view code quality and code security as two distinct issues with different processes and priorities, or is a defect a defect and you try to focus on the most critical or new items first?

Defects are defects, and we do indeed prioritize based on our perception of the impact from a security perspective. Substantial portions of the requested changes in the defect tracking system are feature or portability related and those have traditionally taken a lower priority. With the expanded project team, we are making substantial progress on clearing the backlog of items whilst simultaneously moving forward with the features noted in the roadmap.

## 8. What challenges do you face with regard to maintaining high-quality, secure code that are unique to open source, and how do you overcome those challenges?

The only time people pay serious attention to security software, it seems, is when something breaks—and that means feedback can be terse and so direct-to-the-point that it requires a pretty thick skin to deal with. Fortunately for the project, we also have a core set of users on the mailing list who offer a more balanced view. Our users do also help each other out with responses to items.

## 9. What are your thoughts on the oft-repeated maxim that "given enough eyeballs, all bugs are shallow" discussed here? What could be done to get even more eyeballs (or automated tools) reviewing code?

Reviewing a large code base is an activity that isn't particular exciting and does not generate a lot of focus for the "infinite internet eyeballs" out there. We have always had our entire development history and process out in the open and we welcome constructive feedback from anyone (either privately or publicly). A number of items in our roadmap are focused on making the code base more approachable for those who are working on the source tree or interested in contributing. OpenSSL has been the subject of a large amount of academic research and is certainly the focus of a number of vendors who develop tools, but ultimately it comes down to detailed, careful analysis of specific items.

We are looking forward to the security audit that has recently been announced (sponsored by the Linux Foundation). This will hopefully get to the parts that the many eyeballs don't reach.

"Coverity keeps spotting issues that would go unnoticed otherwise. It's also inspiring our developers to pay more attention to possible NULL dereference and uninitialized values."

**– Trinity Core**

# Interview: Mikko Varpiola, Codenomicon

Codenomicon® provides a suite of next-generation solutions that help forge a better path to total defense. Individually, these solutions provide new layers of security testing, robustness, intelligence, collaboration and security. Codenomicon also offers an array of services that include testing, auditing, training and verification services to aid organizations seeking to advance and enhance their security profile. Codenomicon was recently acquired by Synopsys. We asked Mikko, who has been a cofounder at Codenomicon, to share his thoughts with us.

“We’ve run our code through Coverity Scan, and as a result, we’ve been alerted to potential future security issues within our products. We are grateful to Coverity Scan for this fine service.”

– PowerDNS

## 1. Welcome to the team!

Thank you! We’re thrilled to be joining forces with the Coverity® team and excited about the opportunities for the future.

## 2. If a customer is using Coverity Scan to look at my source code, don’t they already know about vulnerabilities in third party components?

Not necessarily. Many projects incorporate third party components in binary form only, so the source code is not available to Coverity Scan. AppCheck™ performs a static binary scan that essentially X-rays your product to identify contained third party components.

Furthermore, the build process can introduce third party components that are not obvious when examining source code.

## 3. Can you explain using an example?

We recently used AppCheck to examine a popular cross-platform office application suite. The results from the Linux, OS X, and Windows builds had surprising differences, both in their composition of third party components and the known vulnerabilities associated with those components. The table here also shows the highest severity known vulnerability, as measured with the Common Vulnerability Scoring System (CVSS).

| | Linux | OS X | Windows |
|---|---|---|---|
| **Components** | 37 | 30 | 36 |
| **Components with known vulnerabilities** | 7 | 6 | 6 |
| **Known vulnerabilities** | 28 | 28 | 32 |
| **Highest severity vulnerability (CVSS)** | 10 | 10 | 7.5 |

These are the kinds of insights you can get from binary analysis, when AppCheck is looking at the actual executable bits for your product.

## 4. Let's talk about Heartbleed. How did Codenomicon discover Heartbleed?

We were enhancing our Defensics® fuzz testing solution. Specifically, we were adding a feature called SafeGuard to the Defensics TLS test suite.

Fuzzing is a dynamic testing technique in which intentionally malformed messages are sent to target software. If a failure occurs, you know you've found a vulnerability.

Attackers use fuzzing to locate unknown vulnerabilities, then create exploits for found vulnerabilities. You can use fuzzing to proactively defend yourself by locating and remediating vulnerabilities.

In what I'd like to call a 'traditionally applied fuzzing', a fuzzer monitors the target software for failures that result as crashes, busy loops, or resource leakage during testing. The SafeGuard features augments this by checking for behavioral anomalies, or failures that fall outside aforementioned failure modes. For example, we have an authentication bypass check which flags a failure if the fuzzer provides bad credentials but gets authenticated, or is able to proceed past the trust boundary anyhow.

In March 2014, our developers were adding a SafeGuard checks called data leakage, and amplification to our TLS test suite. These check examines the contents, and ratio of outgoing and incoming messages to look for anomalous target behavior. We were using OpenSSL as a reference target and noticed that the SafeGuard was raising a flag for certain malformed TLS heartbeat messages. This was Heartbleed.

"Our code base contains a lot of copy-paste mistakes, so we are glad that there is a tool like Coverity Scan that can detect them. For a human, it would be nearly impossible."

– **OpenTechBFG**

## 5. How can a vulnerability like Heartbleed be avoided?

I have two words for you: better testing. The first step is to manage your software supply chain, which first serves to help you manage known vulnerabilities effectively. When (not if!) a serious vulnerability occurs, you are poised to respond quickly. Part of the problem with Heartbleed was that builders didn't even know which of their products were affected, and buyers didn't know which of their assets were affected.

To manage unknown vulnerabilities effectively, you have to find them. Static source analysis combined with fuzz testing are a powerful combination in the hunt for unknown vulnerabilities.

## 6. What secure coding practices should developers adopt to make their code more secure?

It's really not so much about the developers as the process that surrounds them. You can, and should train developers to write better, secure code, but even the very best developers will make mistakes.

The key is to adopt a secure development process that has appropriate controls in place, and includes supply chain management and unknown vulnerability management.

# Project Snapshots

## C/C++

FreeRADIUS is the most popular open source RADIUS server and the most widely deployed RADIUS server in the world. It supports all common authentication protocols, and the server comes with a PHP-based web user administration tool. It is the basis for many commercial RADIUS products and services, such as embedded systems, RADIUS appliances that support Network Access Control and WiMAX. It supplies the needs of many Fortune-500 companies, telcos and Tier 1 ISPs. It is also widely used in the academic community. The server is fast, feature-rich, modular and scalable. FreeRadius is an open source platform, with its core code written in C++.

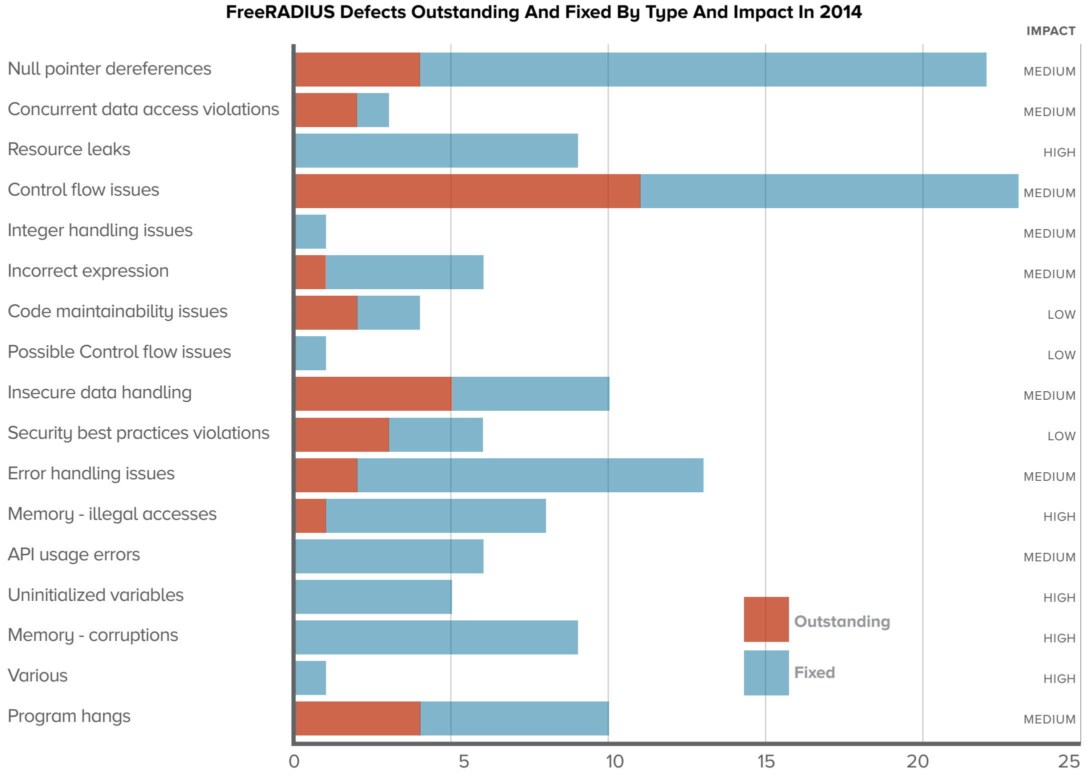
**191,579**
Lines of Code Analyzed

**0.12**
Defect Density

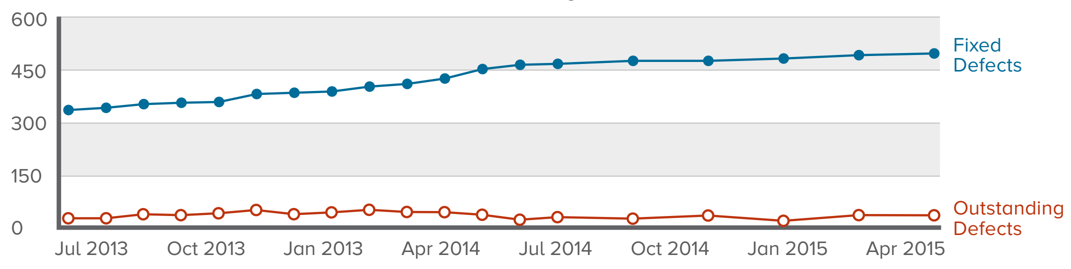
**537**
Total Defects

**486**
Defects Fixed

**FreeRADIUS Defects Outstanding And Fixed By Type And Impact In 2014**



**FreeRADIUS Outstanding vs. Fixed Defects**



## Java

Apache CloudStack is open source software designed to deploy and manage large networks of virtual machines, as a highly available, highly scalable Infrastructure as a Service (IaaS), cloud-computing platform. CloudStack is used by a number of service providers to offer public cloud services, and by many companies to provide an on-premises (private) cloud offering, or as part of a hybrid cloud solution. CloudStack is primarily written in Java and has been using Coverity Scan to improve its quality and security.

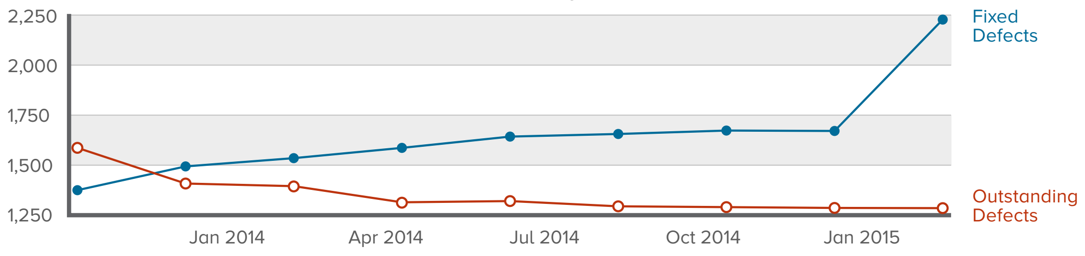
**300,570**
Lines of Code Analyzed

**2.81**
Defect Density

**3,045**
Total Defects

**2,225**
Defects Fixed

**CloudStack Outstanding vs. Fixed Defects**



## C#

Subtitle Edit is a free (open source) editor for video subtitles. It is mainly developed in C#. Subtitle Edit project members have been regularly using Coverity Scan for finding defects in their project.

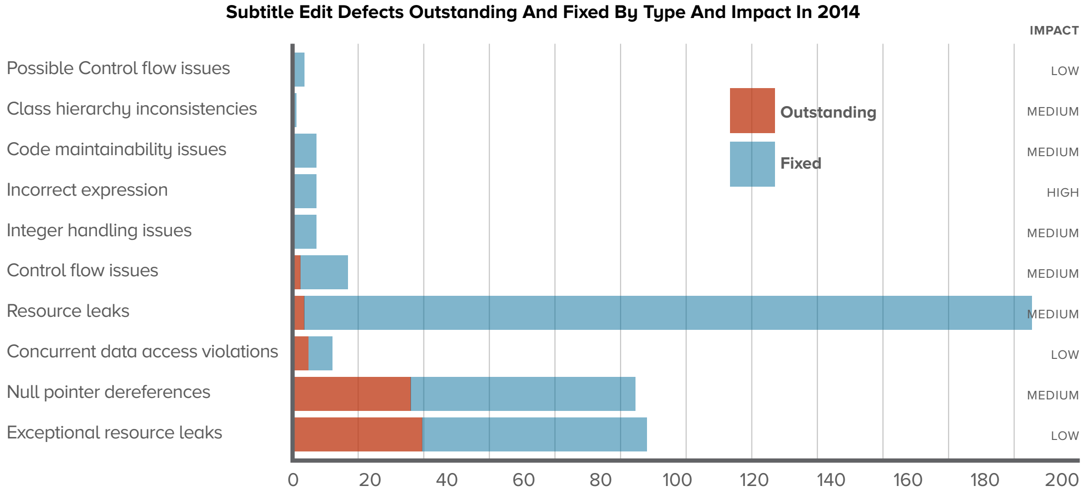
**186,398**
Lines of Code Analyzed

**0.85**
Defect Density

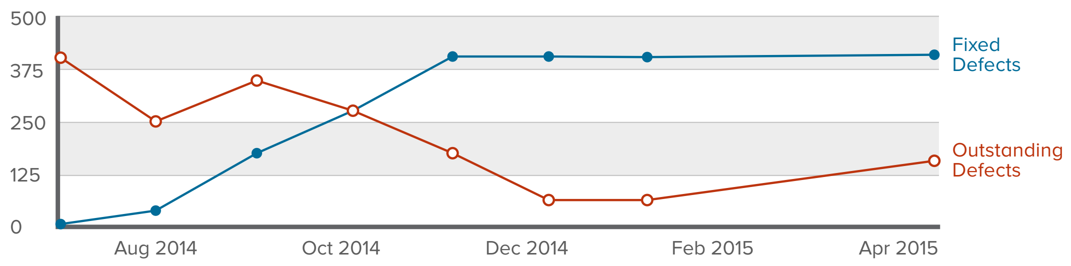
**570**
Total Defects

**406**
Defects Fixed

**Subtitle Edit Defects Outstanding And Fixed By Type And Impact In 2014**



**Subtitle Edit Outstanding vs. Fixed Defects**



## Conclusion

In 2014, the Coverity® Scan service achieved tremendous growth and adoption by open source users, with over 5,100 active projects. This year, analyzing the largest annual sample size to date, the Scan Report identifies several important findings.

- **Commercial code bases are more compliant to security standards than open source in 2014.**

  Based on static analysis defect density, in the 2013 report, we found that open source code outpaced commercial code in quality. This trend continues in 2014; however, this year the report also compared security compliance standards such as OWASP Top 10 and CWE 25, and found that commercial code is more in compliance with these standards than open source.

- **Defect density of both open source software and commercial software has improved since 2013.**

  When comparing overall defect density numbers between 2013 and 2014, the defect density of both open source software and commercial software has improved. Open source defect density improved from 0.66 in 2013 to 0.61 in 2014, while commercial code improved from 0.77 to 0.76.

- **Coverity Scan aided OpenSSL in its post-Heartbleed investigation.**

  According to OpenSSL co-founder Tim Hudson, the Scan service's ability to catch newly discovered issues and highlight the areas where similar code may be found greatly aided the foundation as it coped with the effects of the Heartbleed bug in 2014.

- **Linux remains a benchmark for static analysis defect density.**

  Since joining the Coverity Scan service in 2006, Linux's commitment to quality has remained a key focus. During 2014, Linux leveraged the Scan service to find and fix more than 500 high-impact defects, including resource leaks, memory corruptions and uninitialized variables.

This year's report also makes the assertion that the software industry as a whole must do a better job of balancing security with development speed and feature improvements. Moving forward, developers should use a combination of tools, compliance standards and best-practice principals to produce software that is both high quality and secure—rather than one or the other.

**We would like to thank all of our Scan members and the open source community at large for their interest and support of the Coverity Scan service. If you would like to register a new project, contribute to or observe an existing project, visit us at https://scan.coverity.com/**

## About Coverity Scan

In 2006, the Synopsys **Coverity Scan** service was initiated with the U.S. Department of Homeland Security as a public-private sector research project, focused on open source software quality and security. Coverity Scan now manages the project, providing its development testing technology as a free service to the open source community to help them build quality and security into their software development process. Register your open source project for the Coverity Scan service, and follow us on **Twitter** to get the latest updates.

## About Synopsys

Synopsys, Inc. (Nasdaq: SNPS) is the Silicon to Software™ partner for innovative companies developing the electronic products and software applications we rely on every day. As the world's 15th largest software company, Synopsys has a long history of being a global leader in electronic design automation (EDA) and semiconductor IP, and is also a leader in software quality and security testing with its Coverity® solutions. Whether you're a system-on-chip (SoC) designer creating advanced semiconductors, or a software developer writing applications that require the highest quality and security, Synopsys has the solutions needed to deliver innovative, high-quality, secure products. For more information, visit **www.coverity.com**, follow us on **Twitter** or check out our **blog**.