Comparison of Open Source and Commercial Static Analysis Solutions

Zack Samocha, Senior Director of Products
Coverity

Agenda

- Coverity Scan
- Use case of projects using Scan (Linux, Python, ANTLR)
- Examples of defects
- Java Case Study: Analysis of Jenkins project
- C/C++ Case Study: Analysis of RTOS project
- How to join Coverity Scan

Coverity Scan

Free cloud-based service for the open source community

Proven developer adoption

Coverity founders first published work reported over 500 defects in the Linux kernel

2000

Coverity Scan began

2006

2013

Over 600 projects and 300M lines of code

Over 45,000 defects fixed by the community

coverity®

Coverity Scan how it works

1. Register project
2. Download Coverity "Build" and upload results to Scan.Coverity.Com
3. View/Triage defects

Coverity Scan | Static Analysis

- CPU for Analysis of whole application
- Persistency: Triage, False positive
- Automation: build upload, email
- UI for remediation
- User Management

coverity®

Coverity and Linux

Coverity founders' first published work reported over 500 defects in the Linux kernel

2000

2004

Lines of code:
5.7 million

Defects Identified:
985

Buffer Overflow Defects:
103

False Positive Rate:
20%

2013

Lines of code:
7.6 million

Defects Identified:
4,490

Buffer Overflow Defects:
527

False Positive Rate:
9.7%

Over 18,000 defects identified

Nearly 10,000 defects fixed

coverity®

Linux 3.8 Kernel – Fixed Defects 2013

- 7.8M LOC
- 0.66 Defect density

Category	# defects
API usage errors	11
Code maintainability issues	51
Concurrent data access violations	3
Control flow issues	186
Error handling issues	157
Incorrect expression	72
Insecure data handling	41
Integer handling issues	795
Memory - corruptions	723
Memory - illegal accesses	140
Null pointer dereferences	235
Performance inefficiencies	1
Program hangs	3
Resource leaks	94
Security best practices violations	21
Uninitialized variables	89
Various	1
Total	**2623**

Python Interpreter (C/C++) – Fixed Defects 2013

- 400K LOC
- Defect density : Zero!

Category	#defects
API usage errors	1
Code maintainability issues	1
Control flow issues	12
Error handling issues	22
Insecure data handling	2
Integer handling issues	37
Memory - corruptions	17
Memory - illegal accesses	7
Null pointer dereferences	23
Resource leaks	25
Security best practices violations	6
Uninitialized variables	3
Total	**156**

ANTLR (Java Code) - Fixed Defects 2013

- 40K LOC

Category	Total
Coverity: Exceptional resource leaks	4
FindBugs: Bad practice	3
FindBugs: Dodgy code	6
Coverity: Incorrect expression	1
Coverity: Null pointer dereferences	5
Coverity: Resource leaks	1
Total	**20**

coverity®

Linux- Defect Example Resource Leak

alloc_fn: Calling allocation function "kzalloc". bss_cfg is assigned

```
934
    CID 709078 (#1 of 1): Resource leak (RESOURCE_LEAK)
    alloc_fn: Calling allocation function "kzalloc".
    var_assign: Assigning: "bss_cfg" = storage returned from "kzalloc(132UL, 208U)".
935         bss_cfg = kzalloc(sizeof(struct mwifiex_uap_bss_param), GFP_KERNEL);
    At conditional (1): "!bss_cfg" taking the false branch.
936         if (!bss_cfg)
937                 return -ENOMEM;
938
    noescape: Variable "bss_cfg" is not freed or pointed-to in function "mwifiex_set_sys_config_invalid_data".
939         mwifiex_set_sys_config_invalid_data(bss_cfg);
940
    At conditional (2): "params->beacon_interval" taking the true branch.
941         if (params->beacon_interval)
942                 bss_cfg->beacon_period = params->beacon_interval;
    At conditional (3): "params->dtim_period" taking the true branch.
943         if (params->dtim_period)
944                 bss_cfg->dtim_period = params->dtim_period;
945
```

Linux- Defect Example Resource Leak

bss_cfg is not freed

```
934
    CID 709078 (#1 of 1): Resource leak (RESOURCE_LEAK)
    alloc_fn: Calling allocation function "kzalloc".
    var_assign: Assigning: "bss_cfg" = storage returned from "kzalloc(132UL, 208U)".
935         bss_cfg = kzalloc(sizeof(struct mwifiex_uap_bss_param), GFP_KERNEL);
    At conditional (1): "!bss_cfg" taking the false branch.
936         if (!bss_cfg)
937                 return -ENOMEM;
938
    noescape: Variable "bss_cfg" is not freed or pointed-to in function "mwifiex_set_sys_config_invalid_data".
939         mwifiex_set_sys_config_invalid_data(bss_cfg);
940
    At conditional (2): "params->beacon_interval" taking the true branch.
941         if (params->beacon_interval)
942                 bss_cfg->beacon_period = params->beacon_interval;
    At conditional (3): "params->dtim_period" taking the true branch.
943         if (params->dtim_period)
944                 bss_cfg->dtim_period = params->dtim_period;
945
```

Linux- Defect Example Resource Leak

bss_cfg out of scope and leaks

```
951         switch (params->hidden_ssid) {
952         case NL80211_HIDDEN_SSID_NOT_IN_USE:
953                 bss_cfg->bcast_ssid_ctl = 1;
954                 break;
955         case NL80211_HIDDEN_SSID_ZERO_LEN:
956                 bss_cfg->bcast_ssid_ctl = 0;
957                 break;
    At conditional (6): switch case value "NL80211_HIDDEN_SSID_ZERO_CONTENTS" taking the true branch.
958         case NL80211_HIDDEN_SSID_ZERO_CONTENTS:
959                 /* firmware doesn't support this type of hidden SSID */
960         default:
    leaked_storage: Variable "bss_cfg" going out of scope leaks the storage it points to.
961                 return -EINVAL;
962         }
```

```
> *. CID 709078: Resource leak (RESOURCE_LEAK)
>       - drivers/net/wireless/mwifiex/cfg80211.c, line: 935
> Assigning: "bss_cfg" = storage returned from "kzalloc(132UL, 208U)"
>       - but was not free
> drivers/net/wireless/mwifiex/cfg80211.c:935

Signed-off-by: Bing Zhao <bzhao@marvell.com>
---
 drivers/net/wireless/mwifiex/cfg80211.c |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

diff --git a/drivers/net/wireless/mwifiex/cfg80211.c b/drivers/net/wireless/mwifiex/cfg80211.c
index 3875b1a..6c57e83 100644
--- a/drivers/net/wireless/mwifiex/cfg80211.c
+++ b/drivers/net/wireless/mwifiex/cfg80211.c
@@ -1039,6 +1039,7 @@ static int mwifiex_cfg80211_start_ap(struct wiphy *wiphy,
         case NL80211_HIDDEN_SSID_ZERO_CONTENTS:
                 /* firmware doesn't support this type of hidden SSID */
         default:
+                kfree(bss_cfg);
                 return -EINVAL;
         }
```

[The Fix: http://marc.info/?l=linux-wireless&m=134135643727424&w=2](http://marc.info/?l=linux-wireless&m=134135643727424&w=2)

coverity®

Python Defect - Memory Corruption

Memory was allocated for variable "buffer".
line 10123, memory was de-allocated for variable "buffer".

Show ▾ /Modules/posixmodule.c

This is a historical version of the file displaying the issue before it was in the Fixed state. To see the latest version,

```
10116            length = listxattr(name, buffer, buffer_size);
10117        else
10118            length = llistxattr(name, buffer, buffer_size);
10119        Py_END_ALLOW_THREADS;
10120
     12. Condition "length < 0L", taking true branch
10121        if (length < 0) {
     13. Condition "*__errno_location() == 34", taking true branch
10122            if (errno == ERANGE) {
     14. freed_arg: "free(void *)" frees "buffer".
10123                PyMem_FREE(buffer);
     15. Continuing loop
10124                continue;
10125            }
10126            path_error(&path);
```

coverity®

Python Defect - Memory Corruption

line 10161, memory was again deallocated for variable "buffer", but after NULL check.

```
10153                 start = trace + 1;
10154             }
10155         }
10156     break;
10157     }
10158 exit:
10159     path_cleanup(&path);
          19. Condition "buffer", taking true branch
10160     if (buffer)
          CID 1021198 (#1 of 1): Double free (USE_AFTER_FREE)
          20. double_free: Calling "free(void *)" frees pointer "buffer" which has already been freed.
10161         PyMem_FREE(buffer);
10162     return result;
10163 }
10164
```

coverity®

Python Defect - Memory Corruption

The fix : set buffer to Null after de-allocation

Show ▾ /Modules/posixmodule.c

```
10257            length = llistxattr(name, buffer, buffer_size);
10258        else
10259            length = llistxattr(name, buffer, buffer_size);
10260        Py_END_ALLOW_THREADS;
10261
10262        if (length < 0) {
10263            if (errno == ERANGE) {
10264                PyMem_FREE(buffer);
10265                buffer = NULL;
10266                continue;
10267            }
10268            path_error(&path);
10269            break;
10270        }
10271
10272        result = PyList_New(0);
```

coverity®

ANTLR Defect – Copy Paste Error

This particular defect occurs when a section of code is copied and pasted and the programmer, who intended to rename an identifier, forgot to change one instance

```
88                  }
89
90                  if ( returns!=null ) {
91                          r.retvals = ScopeParser.parseTypedArgList(returns, returns.getText()
92                          r.retvals.type = AttributeDict.DictType.RET;
    original: "r.retvals.ast = returns" looks like the original copy.
93                          r.retvals.ast = returns;
94                  }
95
96                  if ( locals!=null ) {
97                          r.locals = ScopeParser.parseTypedArgList(locals, locals.getText(), g
98                          r.locals.type = AttributeDict.DictType.LOCAL;
    CID 1072510 (#1 of 1): Copy-paste error (COPY_PASTE_ERROR)
    copy_paste_error: "returns" in "r.locals.ast = returns" looks like a copy-paste error. Should it say "locals" instead?
99                          r.locals.ast = returns;
100                 }
101
```

ANTLR Defect – Null Dereferences

"g" is compared against null on line 115, indicating that the developer expects that it might be null. If "g" is actually null, it will be passed to the getStateString() method on line 116, which will throw a NullPointerException on line 133

```
10. Condition "t instanceof org.antlr.v4.runtime.atn.AtomTransition", taking true branch
112                             else if ( t instanceof AtomTransition ) {
113                                     AtomTransition a = (AtomTransition)t;
114                                     String label = a.toString();
    11. Condition "g != null", taking false branch
    12. var_compare_op: Comparing "g" to null implies that "g" might be null.
115                                     if ( g!=null ) label = g.getTokenDisplayName(a.label);
    CID 1072514 (#1 of 2): Dereference after null check (FORWARD_NULL)
    13. var_deref_model: "org.antlr.v4.automata.ATNPrinter.getStateString(org.antlr.v4.runtime.atn.ATNState)" dereferences null "this.g". [hide details]
116                                     buf.append("-").append(label).append("->").append(getStateString(t.target)).append(
```

/tool/src/org/antlr/v4/automata/ATNPrinter.java

```
126         String getStateString(ATNState s) {
127                 int n = s.stateNumber;
128                 String stateStr = "s"+n;
    1. Condition "s instanceof org.antlr.v4.runtime.atn.StarBlockStartState", taking false branch
129                 if ( s instanceof StarBlockStartState ) stateStr = "StarBlockStart_"+n;
    2. Condition "s instanceof org.antlr.v4.runtime.atn.PlusBlockStartState", taking false branch
130                 else if ( s instanceof PlusBlockStartState ) stateStr = "PlusBlockStart_"+n;
    3. Condition "s instanceof org.antlr.v4.runtime.atn.BlockStartState", taking false branch
131                 else if ( s instanceof BlockStartState) stateStr = "BlockStart_"+n;
    4. Condition "s instanceof org.antlr.v4.runtime.atn.BlockEndState", taking false branch
132                 else if ( s instanceof BlockEndState ) stateStr = "BlockEnd_"+n;
    5. Condition "s instanceof org.antlr.v4.runtime.atn.RuleStartState", taking true branch
    6. method_call: Calling method on "g".
133                 else if ( s instanceof RuleStartState) stateStr = "RuleStart_"+g.getRule(s.ruleIndex).name+"_"+n;
134                 else if ( s instanceof RuleStopState ) stateStr = "RuleStop_"+g.getRule(s.ruleIndex).name+"_"+n;
135                 else if ( s instanceof PlusLoopbackState) stateStr = "PlusLoopBack_"+n;
136                 else if ( s instanceof StarLoopbackState) stateStr = "StarLoopBack_"+n;
137                 else if ( s instanceof StarLoopEntryState) stateStr = "StarLoopEntry_"+n;
138                 return stateStr;
139         }
140 }
```

Numerous Options Exist

coverity

FindBugs™
because it's easy

PMD
DON'T SHOOT THE MESSENGER

Clang Static Analyzer

The Clang Static Analyzer is a source code analysis tool that finds bugs in C, C++, and Objective-C programs.

Cppcheck
A tool for static C/C++ code analysis

Considerations

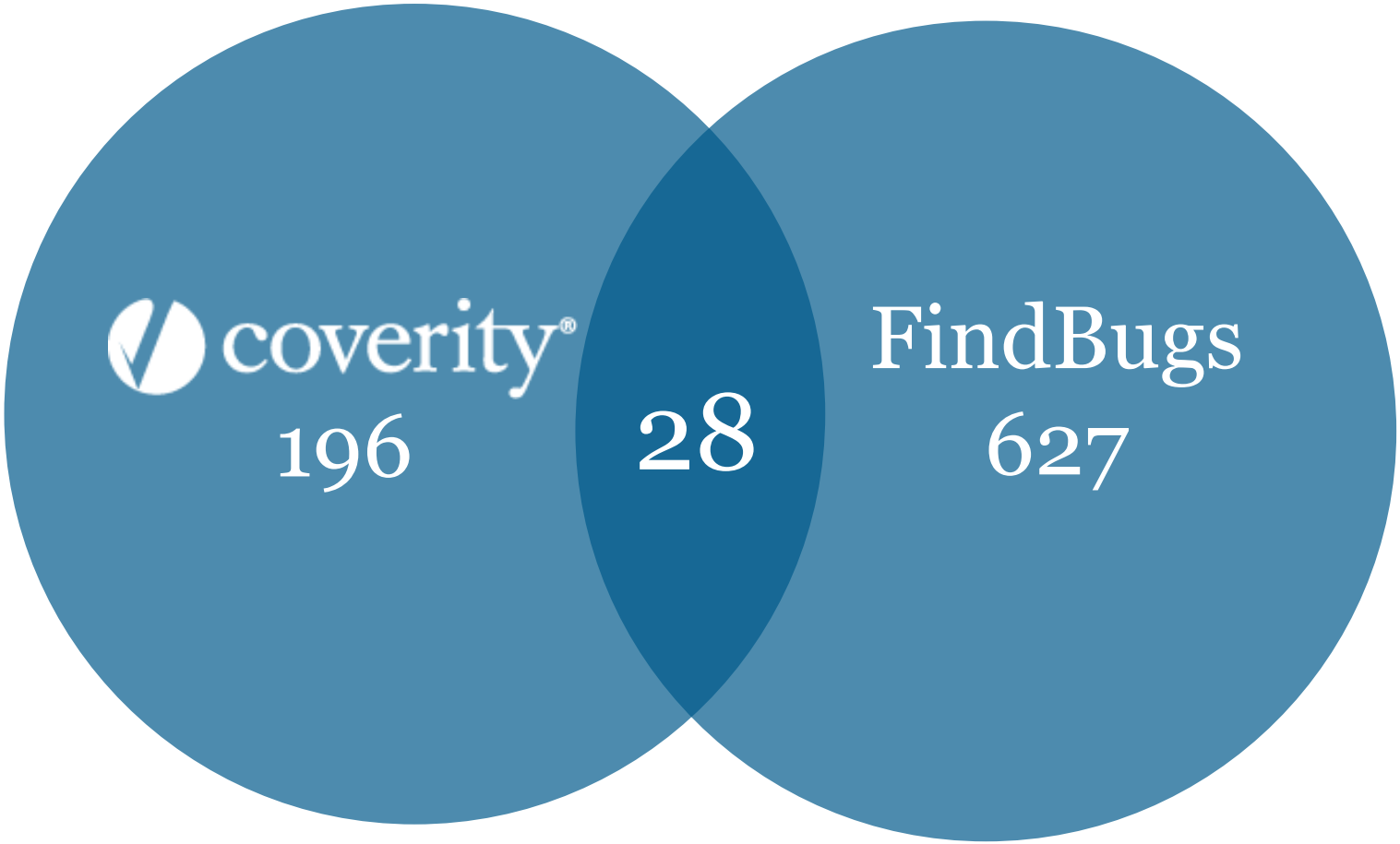
- Does it find critical defects?
- What is the false positive rate?
- Is it actionable?
- Is it accurate?
- Is it integrated to my workflow?
- How do I manage persistency

Jenkins Analysis

- Analyzed Jenkins version 1.496 core code using up-to-date Coverity and FindBugs (as of Dec 2012)

Jenkins

Different Things are Found



Only 28 issues shared between Coverity and FindBugs

coverity®

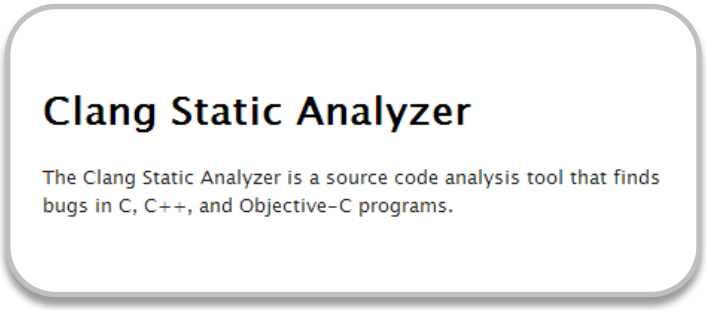
Comparison by Defect Type

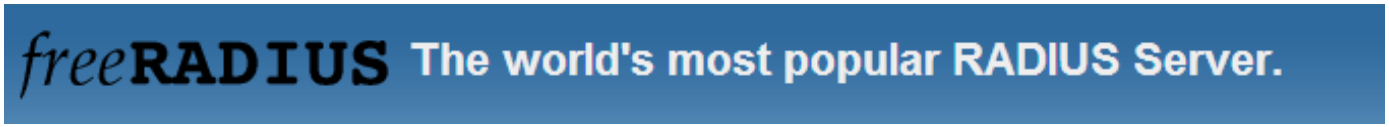
Type	Coverity	FindBugs	Shared Defects
Unhandled exceptions (incl. NULL deref)	79	7	5
Resource leaks	86	12	13
Concurrency problems	22	10	9
Critical Defect Subtotal	187	**29**	**27**
Coding Standards, Best Practices, Other	9	598	1
Total Bugs	196	627	28

coverity®

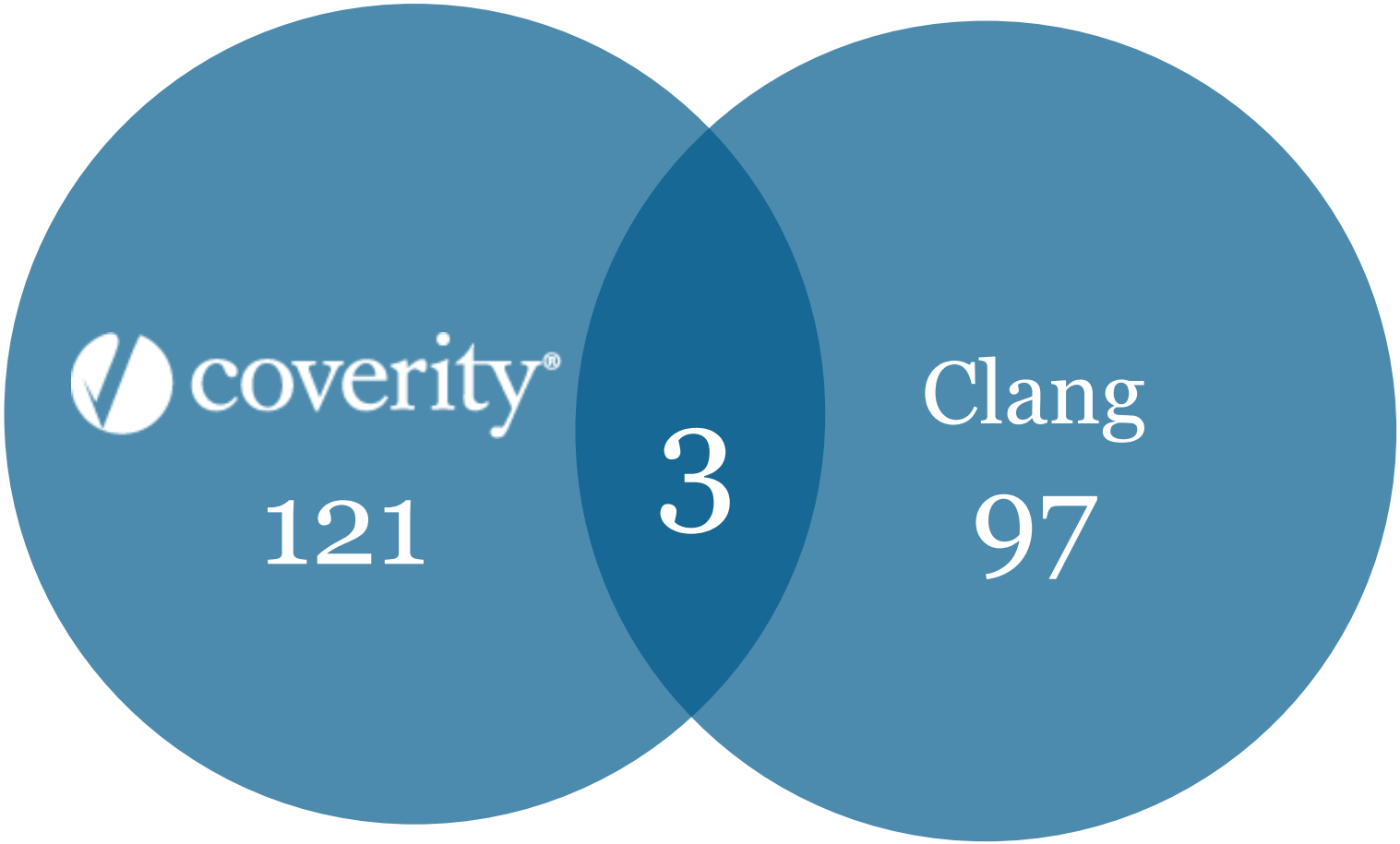
Freeradius Analysis

- Freeradius version: 2.1.12 (released 30th Sep, 2011)
- Clang Analyzer version: checker-275 (23rd May, 2013)
- Coverity version: 6.6.1 (July, 2013)





Different Things are Found



Only 3 issues shared between Coverity and Clang

coverity®

Comparison by Defect Type

Type	Coverity	Clang	Shared Defects
Memory	11	5	0
Resource leaks	9	3	0
Control Flow, Concurrent access, Other	83	30	1
High + Medium Defects	103	**38**	**1**
Coding Standards, Best Practices, Other	9	**59**	2
Total Bugs	121	97	3

coverity®

Freeradius: 2.2.1 (released 17th Sep, 2013)

Security Vulnerability: “We scanned the rlm_eap_tls.c file with the LLVM checker-267, taken from http://clang-analyzer.llvm.org/. It did not find this issue. However, a Coverity scan did discover it.”

http://freeradius.org/security.html

coverity®

Two years later …Freeradius Analysis

- Freeradius version: freeradius 2.2 (released 2013)
- Clang Analyzer version: checker-275 (23rd May, 2013)
- Coverity version: 6.6.1 (July, 2013)

coverity®

Clang Static Analyzer

The Clang Static Analyzer is a source code analysis tool that finds bugs in C, C++, and Objective-C programs.

*free*RADIUS The world's most popular RADIUS Server.

Two years later ….Fixed in FREERADIUS 2.2.

Impact	Category	Coverity found in 2.1.12	Fixed in 2.2.1	Shared	Clang found in 2.1.12	Fixed in 2.2.1	Clang Category
High	Memory - corruptions	7	3				
High	Memory - illegal accesses	4	3		5	3	Use-after-free
High	Resource leaks	9	8		3		Memory leak
High	Uninitialized variables	5	2		2	2	Assigned value is garbage
Medium	API usage errors	6					
Medium	Concurrent data access violations	1					
Medium	Control flow issues	18	5				
Medium	Error handling issues	19					
Medium	Incorrect expression	14	7		9	3	Dead incremeant/ Dead initialization/Unix API
Medium	Insecure data handling	5	5				
Medium	Integer handling issues	1					
Medium	Null pointer dereferences	13	6	1	19	1	Dereference of null pointer
Medium	Program hangs	1					
Low	Code maintainability issues	4		2	59	1	Dead Assignment
Low	Parse warnings	1					
Low	Security best practices violations	13	3				
	Total	121	**42**	3	97	**10**	

42 of Coverity defects were fixed

10 of Clang defects were fixed

FreeRADIUS Quality practices From Alan Dekok

- Use APIs which make it harder for issues to arise (explicit lengths, etc.)
- On 3.0 branch, build with *no* C compiler warnings
- Use autobuilds (https://travis-ci.org/FreeRADIUS/freeradius-server/) builds with clang && gcc, and builds debian packages
- Coverity:
  - New Coverity builds every day, Coverity are emailed to the core team. Many can be fixed directly from the summary in the email
  - This practice ensures basic code sanity. What it *can't* do is ensure logical correctness. We've had a few bugs slip in where the code passes all checks, but is logically incorrect. i.e. it doesn't implement part of a protocol correctly.
  - Finding those issues automatically is harder. Doing a protocol test suite for a complex daemon is very difficult. With the 3.0 branch, we're now running more unit tests, for basic functionality. That helps, but more tests are needed.
  - For us, Coverity is an indispensable part of our daily development routine. It's helped to make FreeRADIUS better software. And it's helped to make us better programmers.

coverity®

How Does Your Code Compare?

The bar has been raised on what is considered *good* quality software

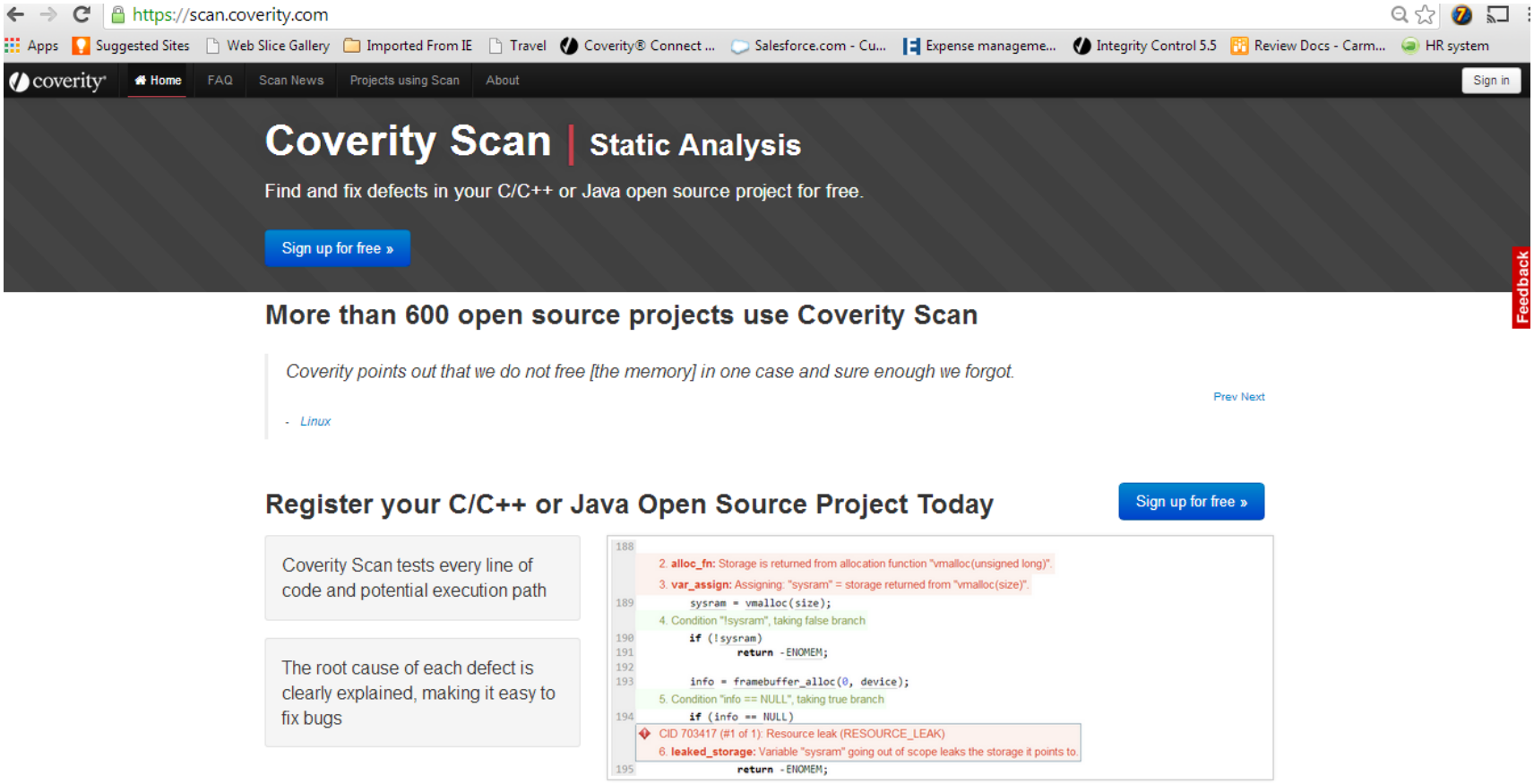
.69 defect density vs. 1.0

Defect Density by Project Size: Open Source vs. Proprietary		
Lines of code	Open Source	Proprietary
<100k	.4	.51
100k-499k	.6	.66
500k-1m	.44	.98
>1M	.75	.66
Average	.69	.68

coverity®

SCAN.Coverity.COM – Free for the Open Source Community

Sign Up Today



coverity®

Q&A