Coverity and LibreOffice

Current Status

Caolán McNamara
Red Hat, Inc.
2014-09-05

Coverity Info

Coverity Blurb

- Stanford Checker research project
  - Commercialized as Coverity in 2002
- Coverity Scan service for “open source” since 2006
  - Free service
- Approx 2,500 projects participating
- Apparently “over 100,000 defects identified by the service have been fixed since the inception of the program”

No bug is too foolish to check for.

http://web.stanford.edu/~engler/BLOC-coverity.pdf

[CVE-2006-0745] X.Org privilege escalation

```
if(getuid() != 0 && geteuid == 0) {
  ErrorF(“only root”);
  exit(1);
}

// all sorts of things only root should be able to do
// geteuid should be geteuid()
```

Scan Process

Build Frequency

- Up to 7 builds per week, with a maximum of 3 builds per day, for projects with fewer than 100K lines of code
- Up to 4 builds per week, with a maximum of 2 builds per day, for projects with 100K to 500K lines of code
- Up to 2 builds per week, with a maximum of 1 build per day, for projects with 500K to 1 million lines of code
- 1 build per week for projects with more than 1 million lines of code
  - ^^^^ That's us there ^^^^

Sizes and Timings

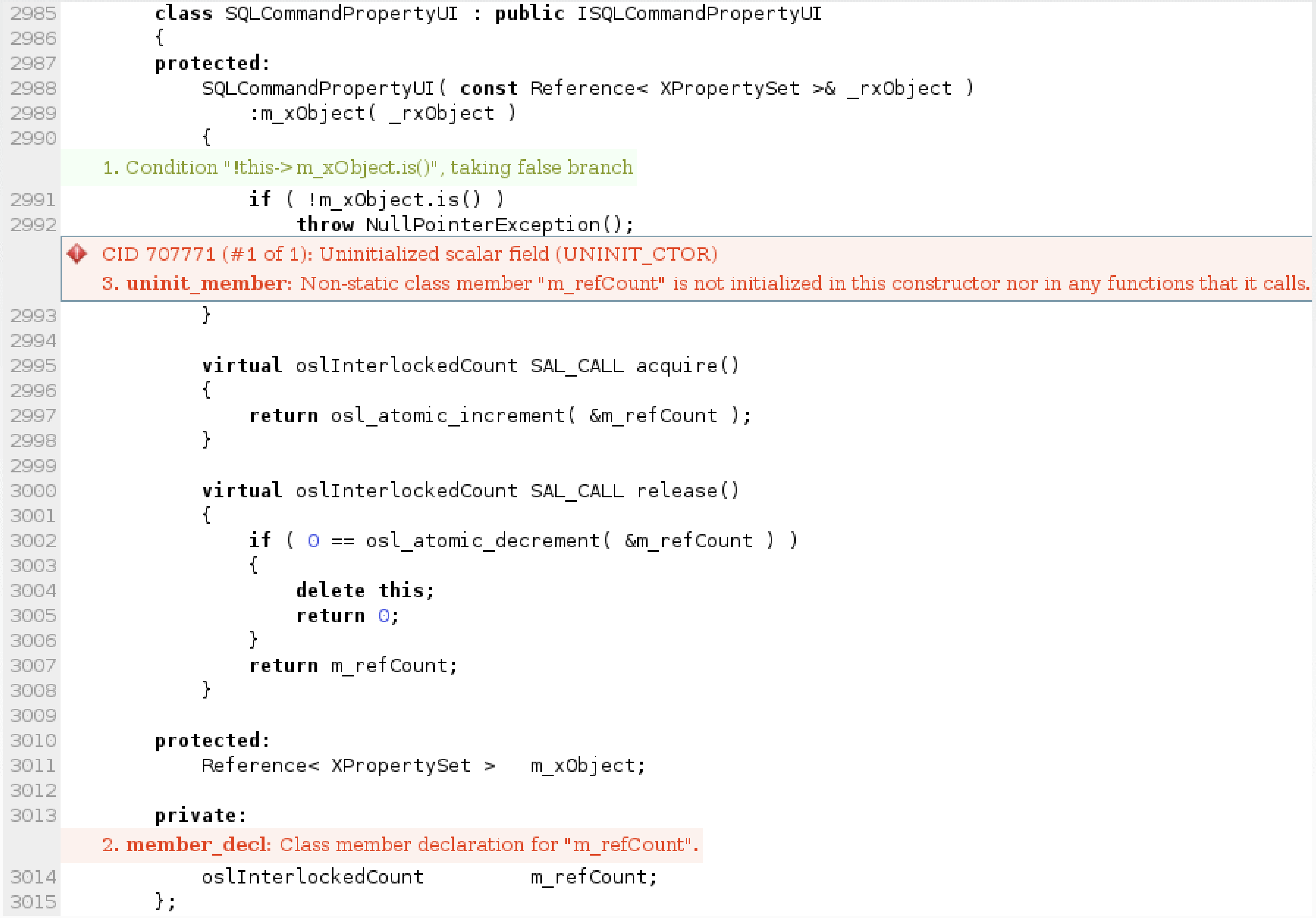
- Sizes
  - 9,478,622 lines of code are built
  - 5,659,730 of those belong to LibreOffice itself
  - 3,779,926(+) in workdir/UnpackedTarball/ (etc)
- Time
  - Cov-scan make on our side takes about 3 to 4 hours
  - Upload of results approx ~1 hour
  - Analysis run on their side 12 to 28 hours

What does it report

UNCAUGHT_EXCEPTION	PASS_BY_VALUE
CHECKED_RETURN	CTOR_DTOR_LEAK
FORWARD_NULL	UNINIT
DIVIDE_BY_ZERO	NO_EFFECT
TAINTED_SCALAR	VIRTUAL_DTOR
DEADCODE	MISMATCHED_ITERATOR
RESOURCE_LEAK	SWAPPED_ARGUMENTS
UNINIT_CTOR	TOCTOU
MISSING_BREAK	COPY_PASTE_ERROR
USE_AFTER_FREE	STRING_OVERFLOW
REVERSE_INULL	SIGN_EXTENSION
MIXED_ENUMS	ARRAY_VS_SINGLETON
NEGATIVE_RETURNS	SIZECHECK
INTEGER_OVERFLOW	SIZEOF_MISMATCH
37 others	

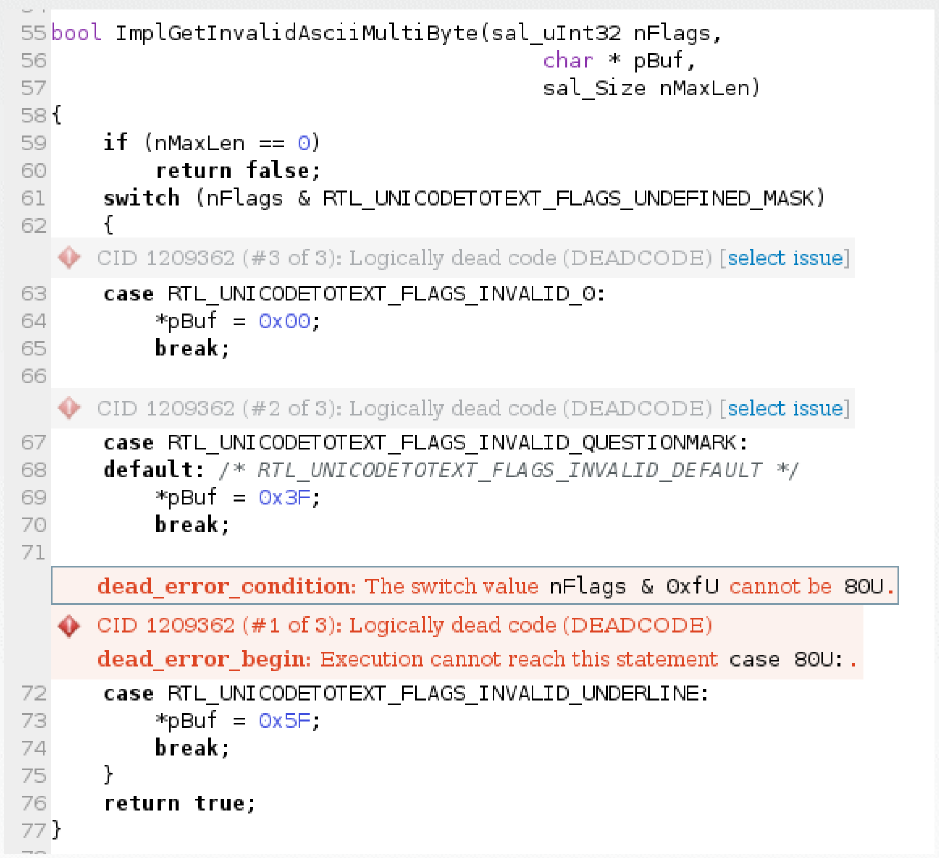
LibreOffice Examples

CID#707771 UNINIT_CTOR



```
class SQLCommandPropertyUI : public ISQLCommandPropertyUI
{
protected:
    SQLCommandPropertyUI( const Reference< XPropertySet >& _rxObject )
        :m_xObject( _rxObject )
    {
1. Condition "!this->m_xObject.is()", taking false branch
        if ( !m_xObject.is() )
            throw NullPointerException();
CID 707771 (#1 of 1): Uninitialized scalar field (UNINIT_CTOR)
3. uninit_member: Non-static class member "m_refCount" is not initialized in this constructor nor in any functions that it calls.
    }

    virtual oslInterlockedCount SAL_CALL acquire()
    {
        return osl_atomic_increment( &m_refCount );
    }

    virtual oslInterlockedCount SAL_CALL release()
    {
        if ( 0 == osl_atomic_decrement( &m_refCount ) )
        {
            delete this;
            return 0;
        }
        return m_refCount;
    }

protected:
    Reference< XPropertySet >   m_xObject;

private:
2. member_decl: Class member declaration for "m_refCount".
    oslInterlockedCount         m_refCount;
};
```

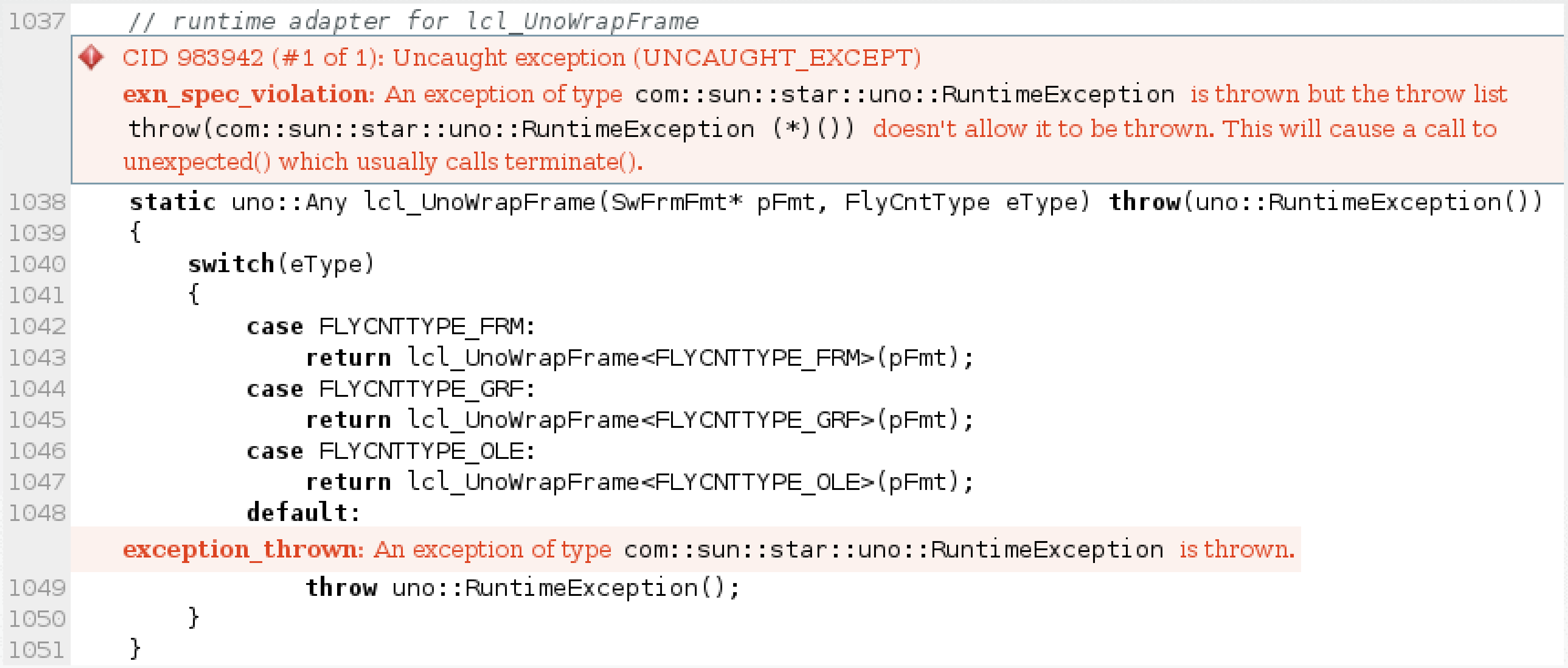
CID#1209362 DEADCODE



```
bool ImplGetInvalidAsciiMultiByte(sal_uInt32 nFlags,
                                  char * pBuf,
                                  sal_Size nMaxLen)
{
    if (nMaxLen == 0)
        return false;
    switch (nFlags & RTL_UNICODETOTEXT_FLAGS_UNDEFINED_MASK)
    {
CID 1209362 (#3 of 3): Logically dead code (DEADCODE) [select issue]
    case RTL_UNICODETOTEXT_FLAGS_INVALID_0:
        *pBuf = 0x00;
        break;

CID 1209362 (#2 of 3): Logically dead code (DEADCODE) [select issue]
    case RTL_UNICODETOTEXT_FLAGS_INVALID_QUESTIONMARK:
    default: /* RTL_UNICODETOTEXT_FLAGS_INVALID_DEFAULT */
        *pBuf = 0x3F;
        break;

dead_error_condition: The switch value nFlags & 0xfU cannot be 80U.
CID 1209362 (#1 of 3): Logically dead code (DEADCODE)
dead_error_begin: Execution cannot reach this statement case 80U: .
    case RTL_UNICODETOTEXT_FLAGS_INVALID_UNDERLINE:
        *pBuf = 0x5F;
        break;
    }
    return true;
}
```

Copy and Paste from previous ImplGetUndefinedAsciiMultiByte without corresponding change of UNDEFINED_MASK to INVALID_MASK

CID#983942 UNCAUGHT_EXCEPT



```
// runtime adapter for lcl_UnoWrapFrame
```

CID 983942 (#1 of 1): Uncaught exception (UNCAUGHT_EXCEPT)
exn_spec_violation: An exception of type `com::sun::star::uno::RuntimeException` is thrown but the throw list `throw(com::sun::star::uno::RuntimeException (*)())` doesn't allow it to be thrown. This will cause a call to unexpected() which usually calls terminate().

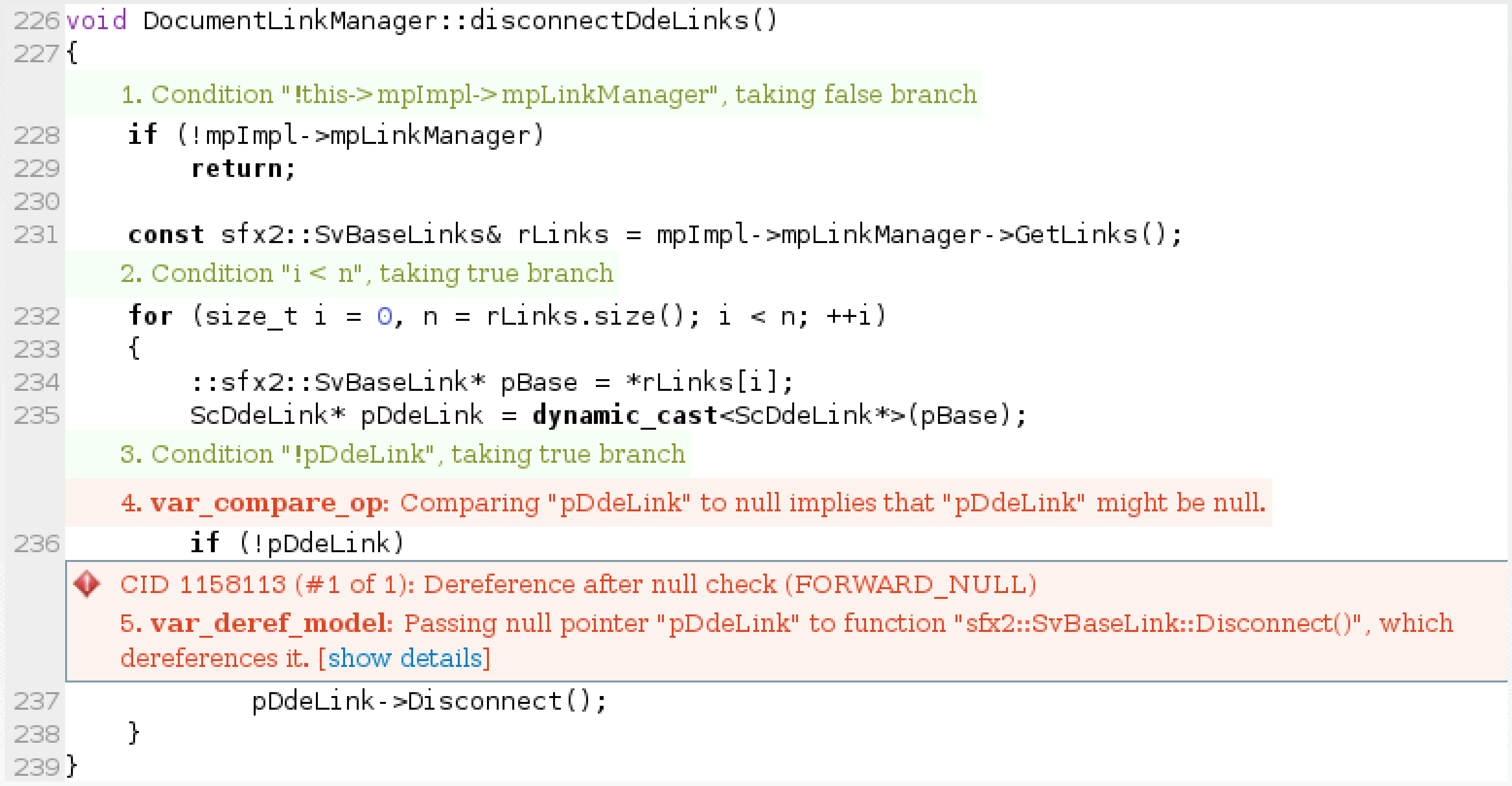
```
static uno::Any lcl_UnoWrapFrame(SwFrmFmt* pFmt, FlyCntType eType) throw(uno::RuntimeException())
{
    switch(eType)
    {
        case FLYCNTTYPE_FRM:
            return lcl_UnoWrapFrame<FLYCNTTYPE_FRM>(pFmt);
        case FLYCNTTYPE_GRF:
            return lcl_UnoWrapFrame<FLYCNTTYPE_GRF>(pFmt);
        case FLYCNTTYPE_OLE:
            return lcl_UnoWrapFrame<FLYCNTTYPE_OLE>(pFmt);
        default:
```

exception_thrown: An exception of type `com::sun::star::uno::RuntimeException` is thrown.

```
            throw uno::RuntimeException();
    }
}
```

That doesn't actually specify what it throws

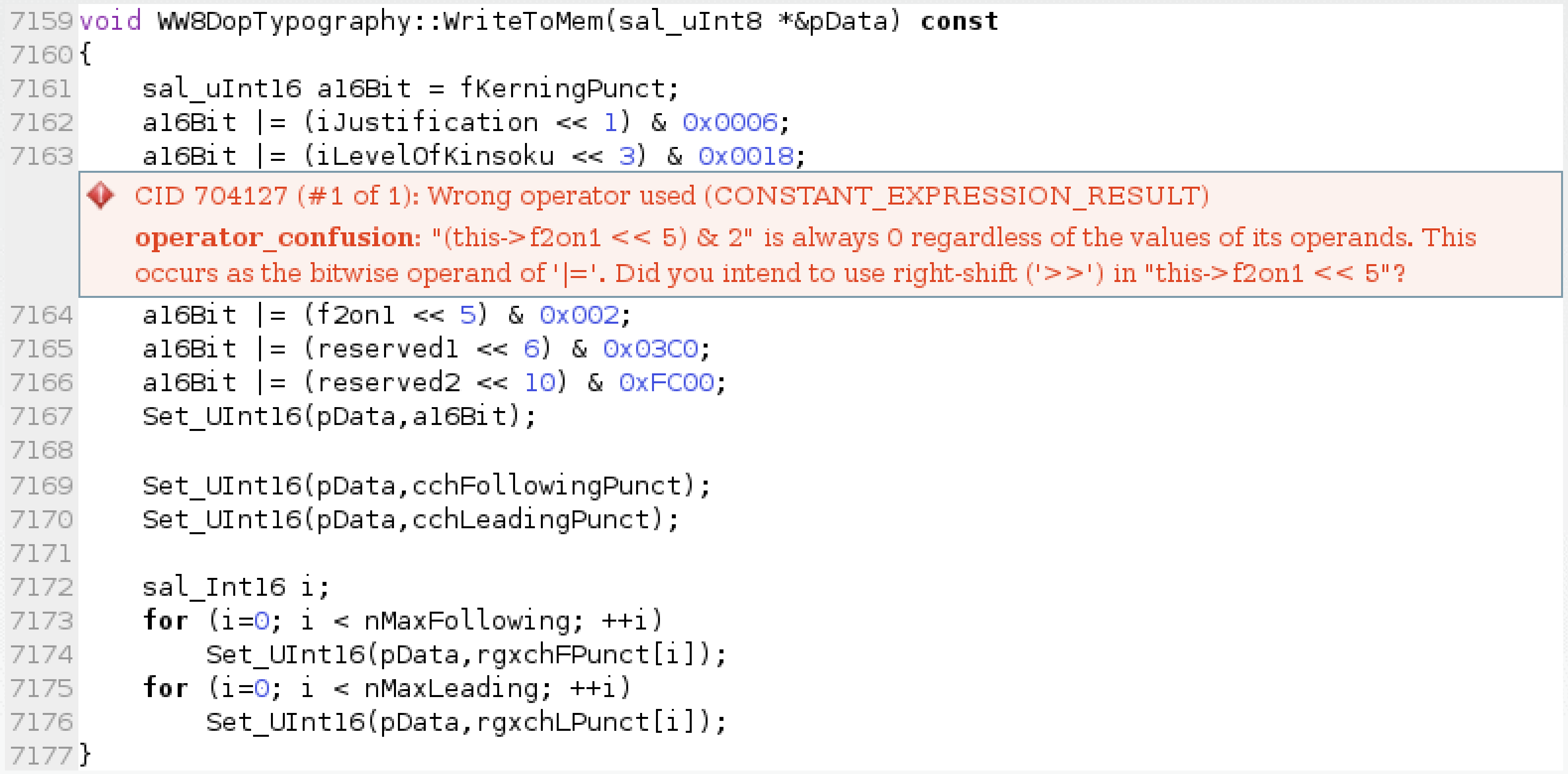
CID#1158113 FORWARD_NULL



```
void DocumentLinkManager::disconnectDdeLinks()
{
    1. Condition "!this->mpImpl->mpLinkManager", taking false branch
    if (!mpImpl->mpLinkManager)
        return;

    const sfx2::SvBaseLinks& rLinks = mpImpl->mpLinkManager->GetLinks();
    2. Condition "i < n", taking true branch
    for (size_t i = 0, n = rLinks.size(); i < n; ++i)
    {
        ::sfx2::SvBaseLink* pBase = *rLinks[i];
        ScDdeLink* pDdeLink = dynamic_cast<ScDdeLink*>(pBase);
    3. Condition "!pDdeLink", taking true branch
    4. var_compare_op: Comparing "pDdeLink" to null implies that "pDdeLink" might be null.
        if (!pDdeLink)
    CID 1158113 (#1 of 1): Dereference after null check (FORWARD_NULL)
    5. var_deref_model: Passing null pointer "pDdeLink" to function "sfx2::SvBaseLink::Disconnect()", which dereferences it. [show details]
            pDdeLink->Disconnect();
    }
}
```

Somebody got confused on checking the result of dynamic_cast

CID#704127 CONSTANT_EXPRESSION_RESULT



```
void WW8DopTypography::WriteToMem(sal_uInt8 *&pData) const
{
    sal_uInt16 a16Bit = fKerningPunct;
    a16Bit |= (iJustification << 1) & 0x0006;
    a16Bit |= (iLevelOfKinsoku << 3) & 0x0018;
```

CID 704127 (#1 of 1): Wrong operator used (CONSTANT_EXPRESSION_RESULT)

operator_confusion: "(this->f2on1 << 5) & 2" is always 0 regardless of the values of its operands. This occurs as the bitwise operand of '|='. Did you intend to use right-shift ('>>') in "this->f2on1 << 5"?

```
    a16Bit |= (f2on1 << 5) & 0x002;
    a16Bit |= (reserved1 << 6) & 0x03C0;
    a16Bit |= (reserved2 << 10) & 0xFC00;
    Set_UInt16(pData,a16Bit);

    Set_UInt16(pData,cchFollowingPunct);
    Set_UInt16(pData,cchLeadingPunct);

    sal_Int16 i;
    for (i=0; i < nMaxFollowing; ++i)
        Set_UInt16(pData,rgxchFPunct[i]);
    for (i=0; i < nMaxLeading; ++i)
        Set_UInt16(pData,rgxchLPunct[i]);
}
```

typo, should be 0x0020 not 0x002, wrong for 14 years

Tips For Developers

MISSING_BREAK

- If the absence of a break is intentional place a comment at the fall-through point
- // fall-through is traditional, but anything will do

```
 case foo:
     ... code ...
+    // fall-through
 case bar:
     ... code ...
     break;
```

Multiple copies of an issue

- In our DEADCODE example there were three instances of the bug
- Also common for UNIT_CTOR that a member is not initialized in multiple constructors
- It is also very common for developers to just fix the first instance and mark the bug as fixed, watch out for that
- Examine the Checkers->All in project->new vs outstanding columns after a new scan.

Unhandled Exceptions #1

- The UNHANDLED_EXCEPTION checked appears to collect what exceptions a method can throw without regard to the input logic

```
void bar(bool bThrow) throw (foo)
{
    if (bThrow)
        throw foo();
}

int foo() throw()
{
    bar(false);
}
```

- Typically I've refactored into a bar_throw, bar_nothrow

Unhandled Exceptions #2

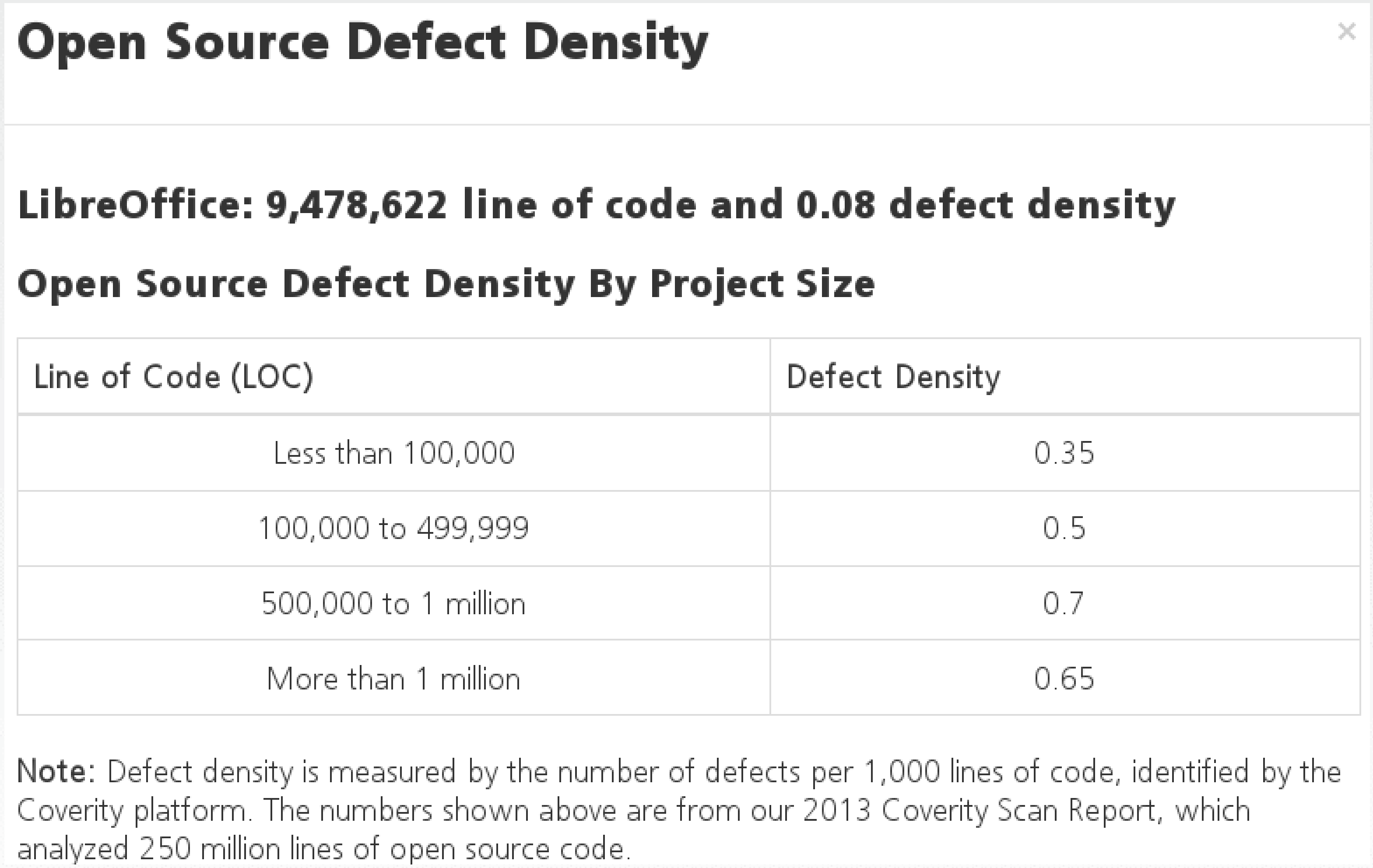
- Coverity appears to examine methods recursively to collect what exceptions callees could throw
- So one missing catch can infect dozens of call-sites and generate multiple cids
- Examine the callstack (window on right) for the site of the exception and work upwards to find the right place to catch it
- One fix can silence multiple cids

Unhandled Exceptions #3

- Derive exceptions from std::exception or uno::RuntimeException
- Don't derive anything from @!&*(^% uno::Exception because all the idl derived signatures have RuntimeException, not Exception
- std::exception derivation also goes for third party libraries, thankfully boost does this

Status

Defect density



Open Source Defect Density

×

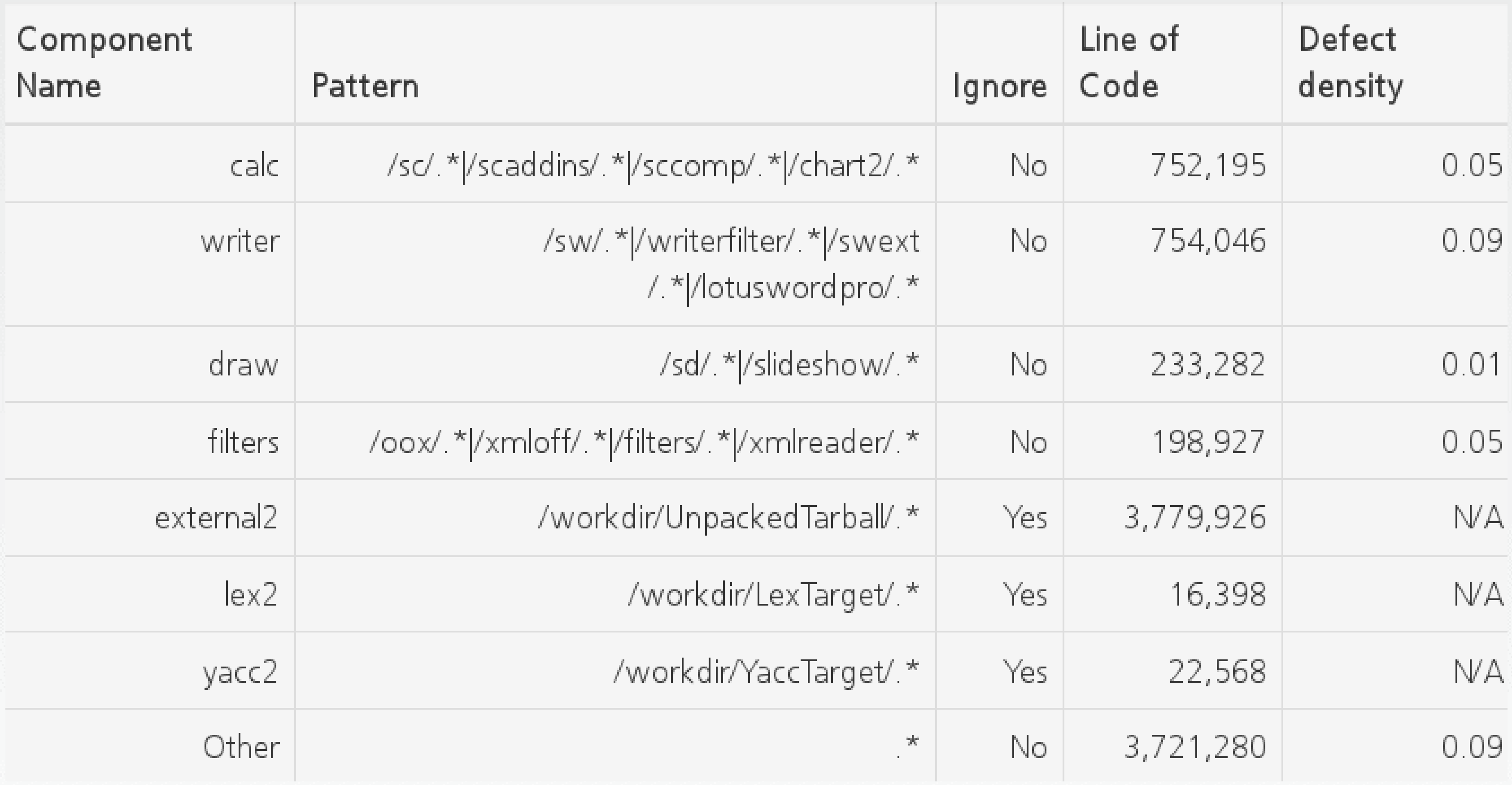
LibreOffice: 9,478,622 line of code and 0.08 defect density

Open Source Defect Density By Project Size

Line of Code (LOC)	Defect Density
Less than 100,000	0.35
100,000 to 499,999	0.5
500,000 to 1 million	0.7
More than 1 million	0.65

Note: Defect density is measured by the number of defects per 1,000 lines of code, identified by the Coverity platform. The numbers shown above are from our 2013 Coverity Scan Report, which analyzed 250 million lines of open source code.

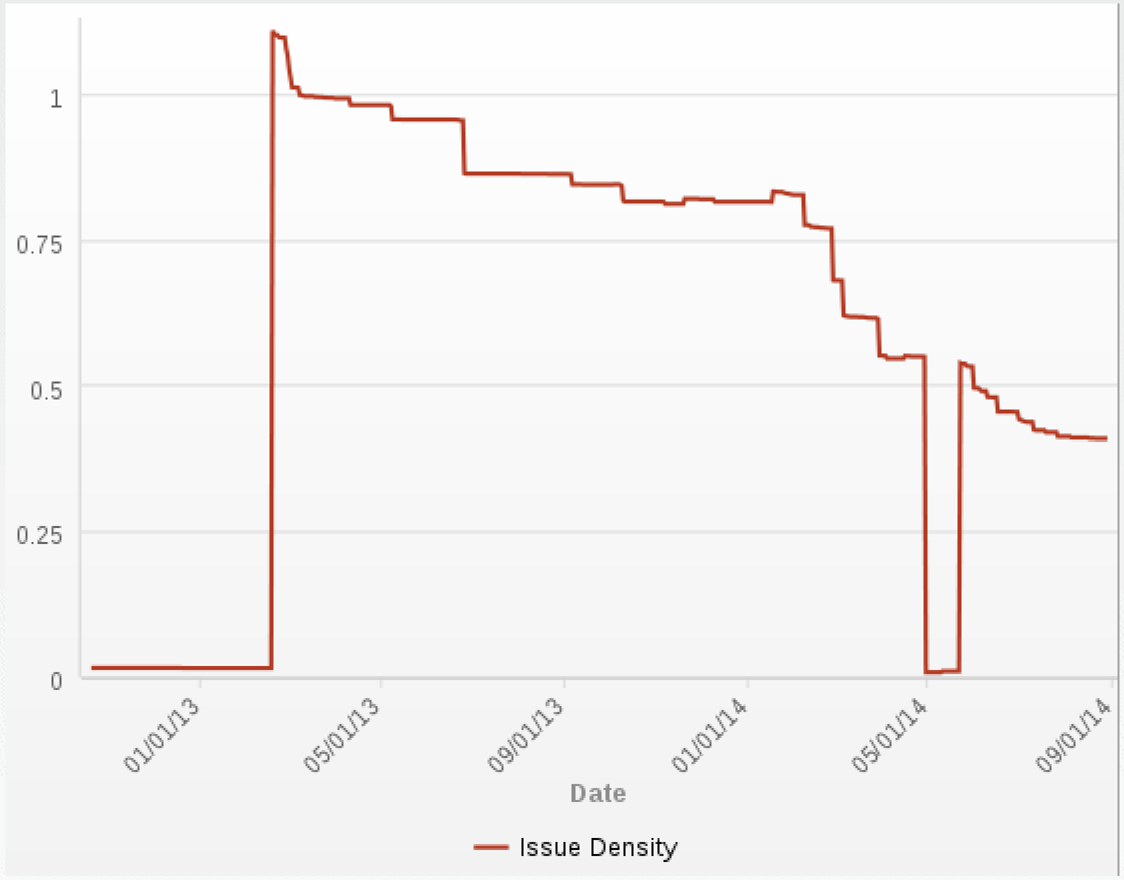
Breakdown



Component Name	Pattern	Ignore	Line of Code	Defect density
calc	/sc/.*|/scaddins/.*|/sccomp/.*|/chart2/.*	No	752,195	0.05
writer	/sw/.*|/writerfilter/.*|/swext /.*|/lotuswordpro/.*	No	754,046	0.09
draw	/sd/.*|/slideshow/.*	No	233,282	0.01
filters	/oox/.*|/xmloff/.*|/filters/.*|/xmlreader/.*	No	198,927	0.05
external2	/workdir/UnpackedTarball/.*	Yes	3,779,926	N/A
lex2	/workdir/LexTarget/.*	Yes	16,398	N/A
yacc2	/workdir/YaccTarget/.*	Yes	22,568	N/A
Other	.*	No	3,721,280	0.09

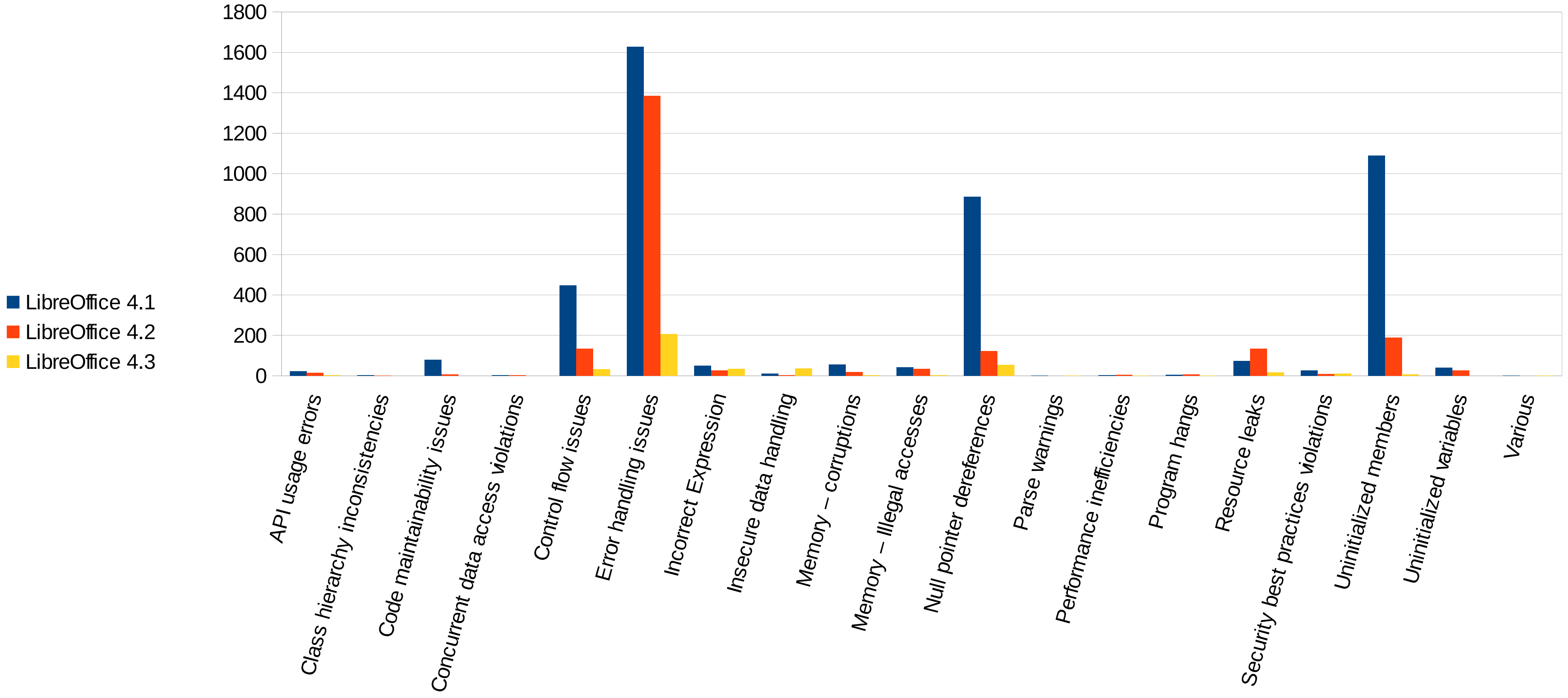
- So 0.08 refers to **OUR** code while (I believe) the ignored third party dependencies are included in the figures under trends->project Lifetime where the total is 0.41

Density over time, included third party code



- Fun blip is using ccache
  - Don't use ccache with cov-scan

Outstanding issues by release by category



What next

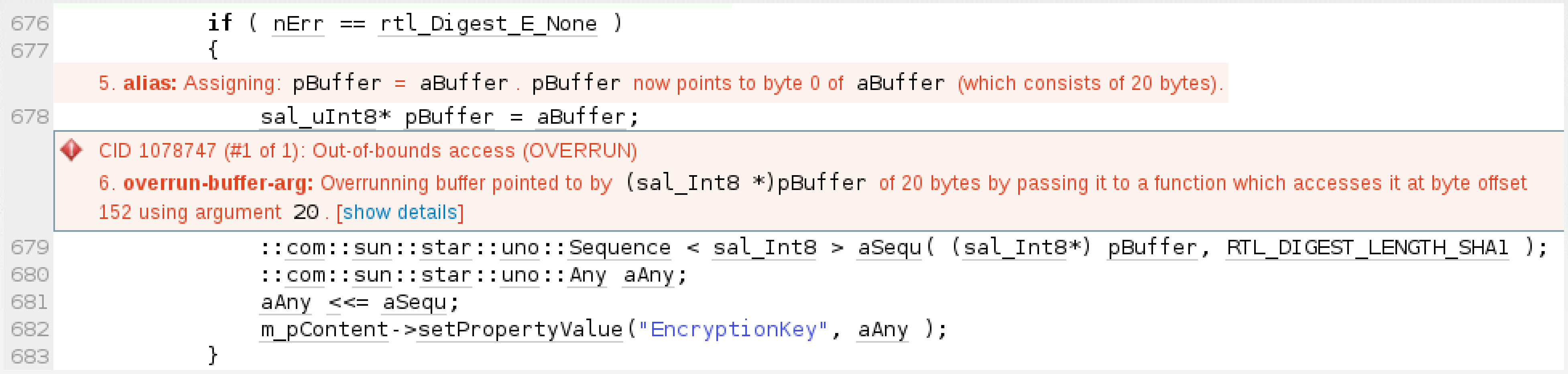
Weekly builds

- Before the massive clean up each refactoring cycle would retain the same problems, but sufficiently differently that coverity saw them as new bugs
- Impossible to distinguish between newly moved old insignificant problem vs truly new significant one.
- Hundreds of *new* per cycle
- Now a handful new per cycle
  - Early new bug detection now possible

Third party libraries

- These are hidden in our view, but account for lots of issues
- Associated projects typically already have coverity scans, libwpd, hunspell, etc
- Non-associated projects ?, icu, firebird, harfbuzz,
  - Help them to help us
- see “external” dir

Final Point: The most annoying false positive



```
        if ( nErr == rtl_Digest_E_None )
        {
5. alias: Assigning: pBuffer = aBuffer . pBuffer now points to byte 0 of aBuffer (which consists of 20 bytes).
            sal_uInt8* pBuffer = aBuffer;
CID 1078747 (#1 of 1): Out-of-bounds access (OVERRUN)
6. overrun-buffer-arg: Overrunning buffer pointed to by (sal_Int8 *)pBuffer of 20 bytes by passing it to a function which accesses it at byte offset 152 using argument 20 . [show details]
            ::com::sun::star::uno::Sequence < sal_Int8 > aSequ( (sal_Int8*) pBuffer, RTL_DIGEST_LENGTH_SHA1 );
            ::com::sun::star::uno::Any aAny;
            aAny <<= aSequ;
            m_pContent->setPropertyValue("EncryptionKey", aAny );
        }
```

- Why does coverity warn about this and how to silence
  - https://communities.coverity.com/thread/2993
- Shows up at multiple call-sites