LibreOffice & Online Securing your documents OWASP - SUPERSEC - Almería 2018

By Michael Meeks

michael.meeks@collabora.com

Twitter: @michael_meeks,

IRC, Skype: mmeeks; +mejmeeks,

"Stand at the crossroads and look; ask for the ancient paths, ask where the good way is, and walk in it, and you will find rest for your souls..." - Jeremiah 6:16

Overview

About / Who / What

Release Processes ...

Free-wheeling
 Open Source LibreOffice

Reactive Security

- Reporting
- Mitre / CVE handling

ProActive Security

- Avoiding problems before they escape.
- Werror, cppcheck
- Coverity Scan
- Core Infrastructure Initiative OSS-FUZZ / American Fuzzy Lop / libFuzzer
- Auditing vs. Fuzzing

About me (us) ...



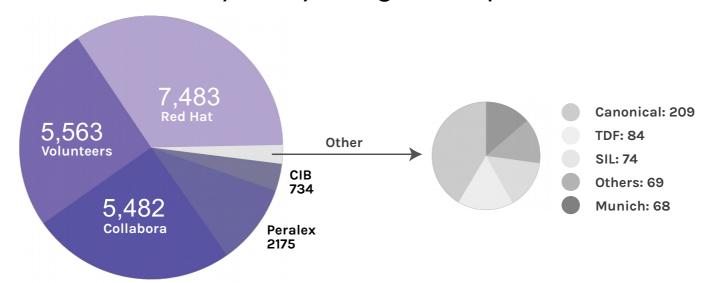
- General Manager of Collabora Productivity
 - Director on The Document Foundation board.
- Previous ~20 years ...
 - Ximian/Novell/SUSE Distinguished Engineer
 - GNOME → shout-out annual conference Almeria July 8th 11th
 - reverse engineering binary file formats for GNOME Office ...
 - working with OpenOffice → LibreOffice since before the beginning
- Interested in security ...
 - Privileged to mentor a Cambridge / Security PhD writing a fuzzer some years back with Ross Anderson.
- Credit where it is due: Caolan McNamara (for RedHat)
 - does the overwhelming majority of the heavy lifting described herein.



What is LibreOffice?

A FOSS project ...

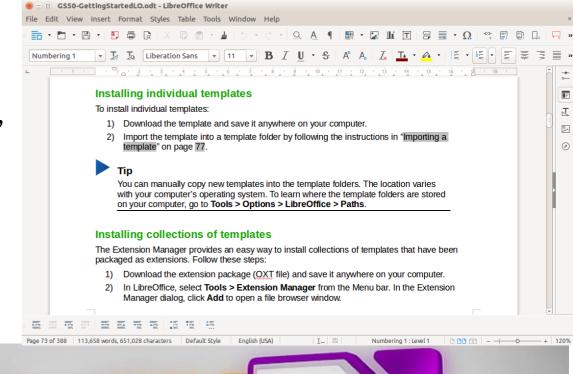
- ~300 hackers each year contributing code changes
- ~1000 total dev community: translators, docs, QA, UX, etc.
- Anyone can push code to LibreOffice via gerrit (easy starter hacks)
- Who is involved? Commits by affiliation:
- who handles security heavy lifting? Enterprise distributions.



What is LibreOffice?

A powerful, interoperable, Office Productivity Suite ...

- Cross-platform: Linux, Mac, Windows, Android
- Online → in your web browser.





LibreOffice Challenges - complexity

Size

- Estimated ~200 million users
- ~6 million lines of code
- Best clean build times ~1 hour ... product builds slower.

Speed of development a year:

• ~16,000 code commits, and roughly:

```
46,168 files changed, 1,176,032 insertions(+), 1,120,754 deletions(-)
```

- Cost of auditing 1m LOC @ 500 LOC/hour
 - 2,000 hours work (per year)

LibreOffice Challenges - Legacy & Platforms

Legacy ... 30+ years of goodness

- StarOffice:
 - 1985 Zilog Z80 ...

 Pre-dates MS Office.
 - OS2, MacOS / PPC
 - DOS, Windows 16bit



Wikipedia / Masterhit

Currently have support for:

- OS/X, Windows +64bit, Android (ARM+Intel),
- Linux: Intel, ARM+64, PowerPC+64, Itanium, Sparc, S390, Alpha
- AIX/PPC, Haiku, iOS/ARM,

LibreOffice Challenges - File filters ...

We support a huge number of legacy binary formats

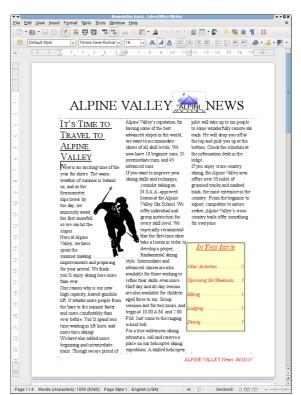


- 175+ import filters: ...
 - Visio, Wordperfect, Quark Express, Publisher etc. ...

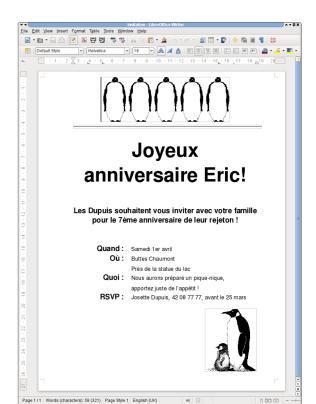
AppleWorks 6.0



Mac Write Pro 1.5



Write Now 4.0



LibreOffice Challenges - Scheduling

Release scheduling

- Three branches
 - git master daily snapshots, sometimes Alphas.
 - Fresh ~monthly minor, 6 monthly major
 - Stable ~monthly minor, inherits Fresh 6 monthly.
 - Fresh & Stable are interleaved for feasibility
- Enterprise Long Term Supported versions

Somewhat similar code-base project

- Co-ordination on officesecurity@freedesktop.org
- Incredible approach to embargo: Feb 14th suggestions etc.

LibreOffice Challenges - Resourcing

Security is one important property of software

- Internationalization
- Accessibility
- Performance / Memory use
- Platform & toolkit churn / bit-rot Uniscribe → Harfbuzz etc.
- Hardware evolution → parallelism
- Language churn Java, python, perl, XSLT, rust(?) ...
- Standard language feature use eg. C++ templates
- Competitive Feature set
- Interoperability
- Quality → low regression & overall bug count ...

Reactive Security

Reactive Security - Mitre/CVE

Issues are reported to officesecurity@freedesktop.org

- GPG keys available for most sensitive issues..
- Detailed here:
 - https://www.libreoffice.org/about-us/security/advisories/ & by vendors.
- All users of the code-base & derivatives represented there

Mitre / CVEs – hello?...

- The requests: mid 2011 Stephen Coley then cve@mitre.org

 The LibreOffice project ... is blessed with an abnormally large number of vulnerabilities, which we are fixing rapidly. ... Is it possible to get a chunk of CVE identifiers as an upstream project to hand out and manage for LibreOffice? [preferably without too much overhead]. ... Advice much appreciated.
- Reply came there none … too scary to let us file them?

CVE flow ...

The reality today

- Individuals now file CVEs and publish them without notifying us
 - Sometimes for oss-fuzz issues that never escaped to the public, that they neither found, nor fixed → huh?
 - Said individuals appear to be anonymous → not glory hounds (?)
- Disclosure
 - Our code is public, all our flaws are already disclosed.
 - Of course finding them can be hard … we'll see later
 - We ask for embargos to match our staggered release process.
- Mitre / CVE brand is still treated seriously by many ... needless fire-drills ...

We prefer a constructive & relational approach to bug reporting & fixing.

This is a flow process ... don't break the flow except in emergency.

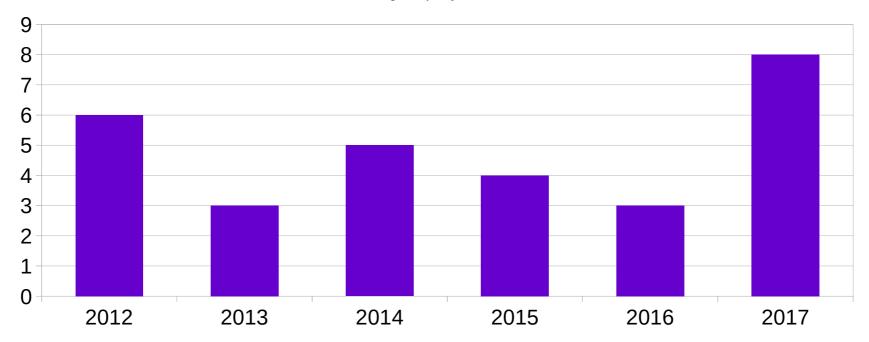
How many reports / CVEs do we get?

Some stats

 The norm is that by the time the CVE is publicised, people are protected – and development continues.

LibreOffice CVEs per year

Including 3rd party advisories.



Pro-Active Security.

Basic improvements ...

Warnings:

- gcc: -Wall -Wextra -Wendif-labels -Wundef -Wunused-macros
- -fmessage-length=0 -fno-common -pipe
 - core code now compiles cleanly

cppcheck linting

- 1600+ cleanup commits thanks to Julien Nabet, Jochen Nitschke & others
- Clang plugins thanks to Noel Grandin & others
 - ~100 of these subsetting C++ adding checks and enforcing good practice.
 - big std::unique_ptr cleanups to avoid memory blow-outs with exceptions ...
- Code review for all back-ports.
- API improvements to kill undefined ... '<<'

Coverity: static checking ...

Great static checker

- Continuously adding new tests & running vs. the code-base
- Huge suite of buffer-overflow, tainted data, bounds-checking etc. etc. etc. tests:
- Hard data on Open Source vs Proprietary. ~1bn lines scanned

Size of Codebase (Lines of Code)	Open Source Code	Proprietary Code
Lines of Code	252,010,313	684,318,640
Number of Projects	741	493
Average Project Size (lines of code)	340,094	1,388,070
Defects Outstanding as of 12/31/13	149,597	492,578
Defects Fixed in 2013	44,641	783,799
Defect Density	.59	.72

Coverity Scan: LibreOffice

Project Name LibreOffice
Lines of code analyzed 6,374,143
On Coverity Scan since Oct 09, 2012
Last build analyzed 1 day ago

Language C/C++ Secondary Language Java

Repository URL http://cgit.freedesktop.org/libreoffice/core

Homepage URL www.libreoffice.org

License MPL (Mozilla Public License)

Want to view defects or help fix defects?

◆ Add me to project

Analysis Metrics

Version: 2017-12-17

Dec 18, 2017

Last Analyzed

6,374,143

Lines of Code Analyzed

6,340,306

Lines of Code in Selected Components 0.00

Defect Density

Would you like to view the project defects, or help improve its quality by fixing defects?

+ Add me to project

Defect changes since previous build dated Dec 15, 2017

4

Newly detected

9

Eliminated

Never quite zero - but rounds down.

Coverity Scan - results ...

An awesome contribution from Coverity.

Some Java & other false-positivies,

More useful - the weekly E-mails with deltas: what changed ... eg.

Please find the latest report on new defect(s) introduced to LibreOffice found with Coverity Scan.

11 new defect(s) introduced to LibreOffice found with Coverity Scan.

8 defect(s), reported by Coverity Scan earlier, were **marked fixed** in the recent build analyzed by Coverity Scan.

Sample feedback:

```
*** CID 1435443: API usage errors (SWAPPED_ARGUMENTS)
/svx/source/accessibility/svxrectctaccessiblecontext.cxx:
854
   in RectCtlAccessibleContext::FireChildFocus(RectPoint)()
        CID 1435443: API usage errors (SWAPPED_ARGUMENTS)
>>>
        The positions of arguments in the call to
>>>
"NotifyAccessibleEvent" do not match the ordering of the
parameters:
   * "aNew" is passed to "_rOldValue"
   * "aOld" is passed to "_rNewValue"
[line] 854
NotifyAccessibleEvent(AccessibleEventId::STATE_CHANGED, aNew, aOld);
```

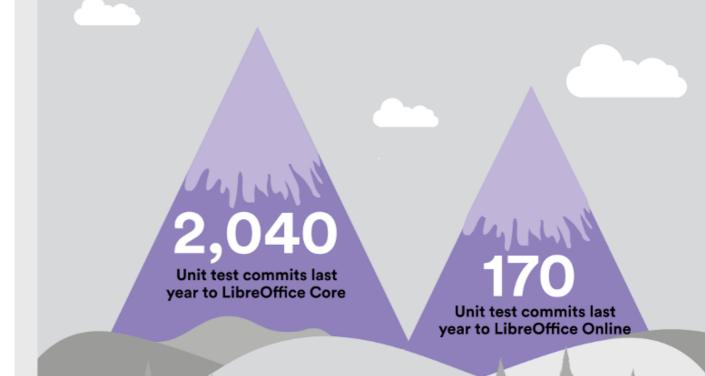
Sample feedback: Caolan ~instant fixes ...

```
*** CID 1435442: Error handling issues (CHECKED_RETURN)
/vcl/source/image/ImplImageTree.cxx: 611 in
ImplImageTree::getNameAccess()()
605
606
            return rNameAccess.is();
607
608
609
        css::uno::Reference<css::container::XNameAccess> const &
ImplImageTree::getNameAccess()
610
        CID 1435442: Error handling issues (CHECKED_RETURN)
>>>
        Calling "checkPathAccess" without checking return value
>>>
        (as is done elsewhere 4 out of 5 times).
611
            checkPathAccess();
612
            return getCurrentIconSet().maNameAccess;
613
        }
```

Security: Unit tests Keeping bugs fixed

- One of our first investments: create a unit-test framework.
- First file-based tests: previous CVE documents
 - Oh dear ~50% regressed
- Now: we have systematic testing of CVE and other problem documents in every build.
- Took this idea & expanded it ...

Annual unit test creation



Loop: Load, Export & Validate - ~100k files ...

Files scraped from every available public bugzilla eg.

- TDF, Launchpad (some), Freedesktop, Mozilla, GNOME, KDE, Gentoo, Mandriva, Novell, AbiSource, W3C SVG test archives
 - bin/get-bugzilla-attachments-by-mimetype the more the merrier ...
- Ideal documents ie. known 'bad'
- If you file your bug, and attach a document we keep it loading & saving

Keep around ~zero Import/Export failures vs. master

finds bugs that fuzzers often find shortly afterwards ...

With Sanitizers

- Runs regularly use Clang / UbiSan (used to use valgrind)
- Finds 'interesting' threading & other less deterministic issues ...

Core Infrastructure Initiative: OSS-Fuzz

Core Infastructure Initiative



Setup in the aftermath of the SSL / Heartbleed bug.

Amazon Web Services, Cisco, Dell, Facebook, Fujitsu, Google, IBM, Intel, Microsoft, NetApp, Rackspace, VMware and The Linux Foundation Form New Initiative to Support Critical Open Source Projects

By Linux Foundation | April 24, 2014 | Press Release

- Huge Testing infrastructure provided by Google
 - Used for Chrome & many other OSS projects.
 - We were an early adopter: already using AFL.
 - ~1000 core cluster to hugely accelerate testing.
- Significant RedHat leadership & investment here too.

What fuzzing do we use:

Lots of goodness: ~50 fuzz targets & using:

- libFuzzer (the default fuzzer engine)
- afl fuzzer engine
- In combination with
 - address sanitizer (asan, the default)
 - undefined behaviour sanitizer (ubsan) enabled.
- Google's generous resource investment keeps us ahead.

Document Liberation - ~70 fuzz targets

- Used for more obscure file formats.
- Also heavy OSS-Fuzz users.

American Fuzzy Lop (a Rabbit)

Interesting work here

- Built on top of Clang.
- "Instrumentation-guided, genetic fuzzer capable of synthesizing complex file semantics in a wide range of non-trivial targets, lessening the need for purposebuilt, syntax-aware tools"
 - It watches the code and breeds badness.
- Catches new bugs on master rapidly.
- Catches assertions too ...

Seed Corpus

- Automatically condensed from 100k docs:
 - http://dev-www.libreoffice.org/corpus/



LibFuzzer +1 coverage guided fuzzing

Another LLVM tool

Inspired by AFL: same same but different ...



Similar idea - breed your Corpus

- A set of helpful sample, minimal documents / data
- Combine these in interesting ways and feed them into the code
- Watch what the code does: do we get more coverage?
- If so insert it back in the corpus & mutate / breed from
- Occasionally minimize / condense the corpus while retaining the code coverge
- Share corpus with AFL eg.

Despire similar inspiration - finds different bugs ...

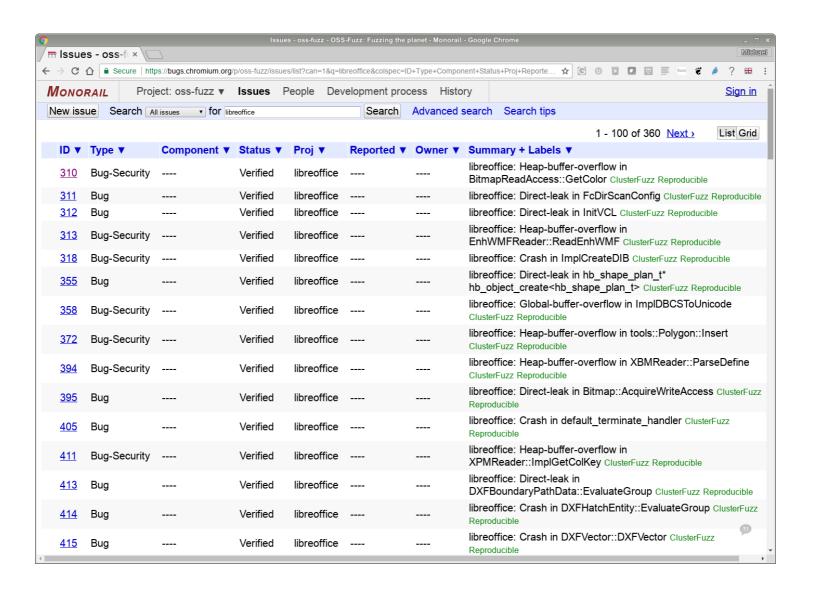
Automatic test case reduction ...

Smarts applied to test case redux too

- Taking a giant / tangled file and intelligently shrinking it while keeping the crash.
- Exciting to see some big HTML file shrunk down to:

 - sw/qa/core/data/html/fail/ofz5909-1.html 95 bytes ofz#5909 Null-dereference READ
 - <div id="sdfootnote1"></d>

OSS-Fuzz dashboard:



Forcepoint ...



Generously donating their expertise

- Another proprietary fuzzer ...
 - Getting some torture testing from our code.
 - thanks to Antti Levomäki and Christian Jalio

You might think that all the problems are already found / fixed ...

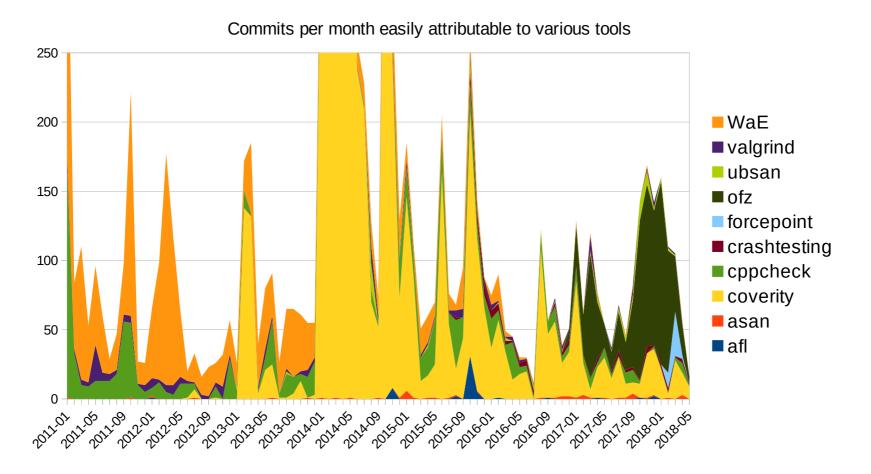
A number of interesting new issues from their work

- 39 new issues fixed.
 - crashers, leaks, missing exception handling
- New strategies find new things → then diminishing returns ...

Fuzzing - the take home ...

New tools find new bugs - and over time that reduces

 Hard to see – not everyone uses consistent git commit tooling references, eg. crashtesting is badly under-represented.



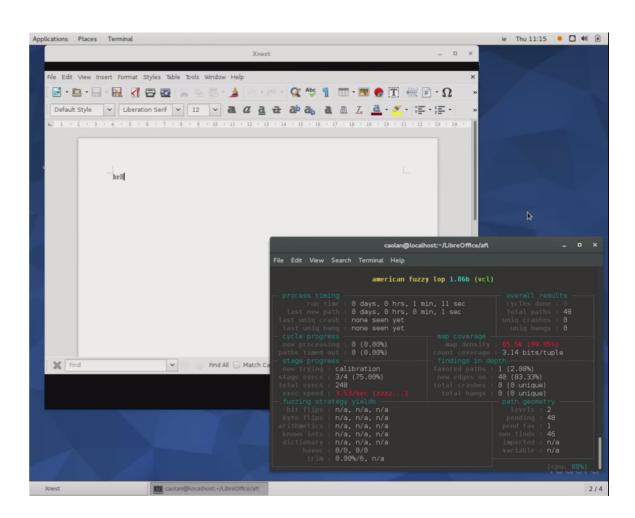
Fuzzing - for User Input ...

An extraordinary use of fuzzing - to drive the Keyboard/mouse

- http://caolanm.blogspot.com.es/2015/10/finding-ui-crashes-by-fuzzing-input.html
- Typing into the suite
- Found a ~dozen bugs
- some long standing evil bugs.

eg. timer race undoing impress slide insertion caused crash.

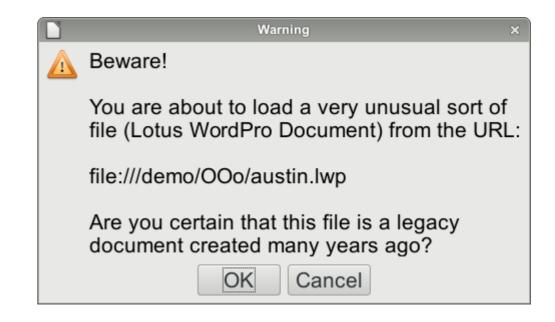
Now so fast it can't be seen working ...



Better controlling the attack surface

Exotic Filter Annotation

- Recently added some context.
- A configurable compile option.
- Its a great thing to be a Swiss Army Knife of formats



ADMX / Sysadmin lock-down / disable per-filter

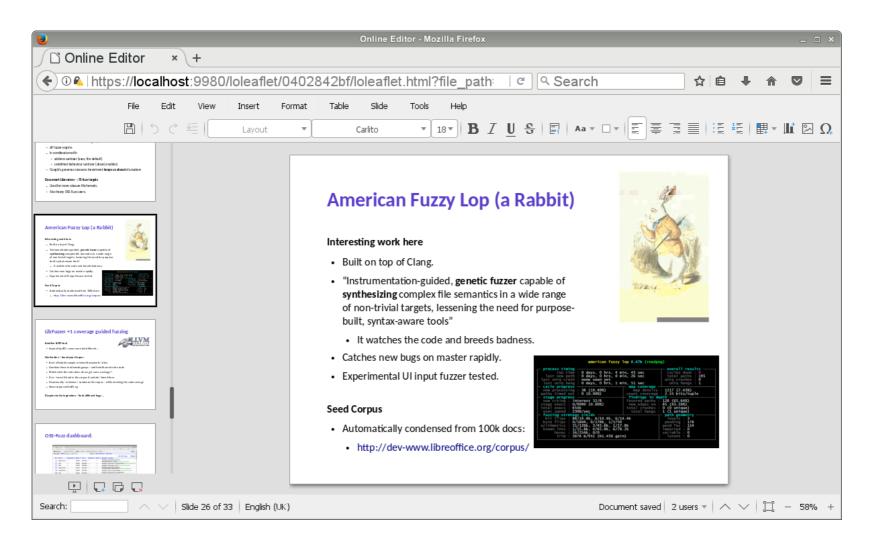
Competition / Other options

- Retro-fitted layered binary validator
- disable older binary filters by default + "safe mode"

Online Security.

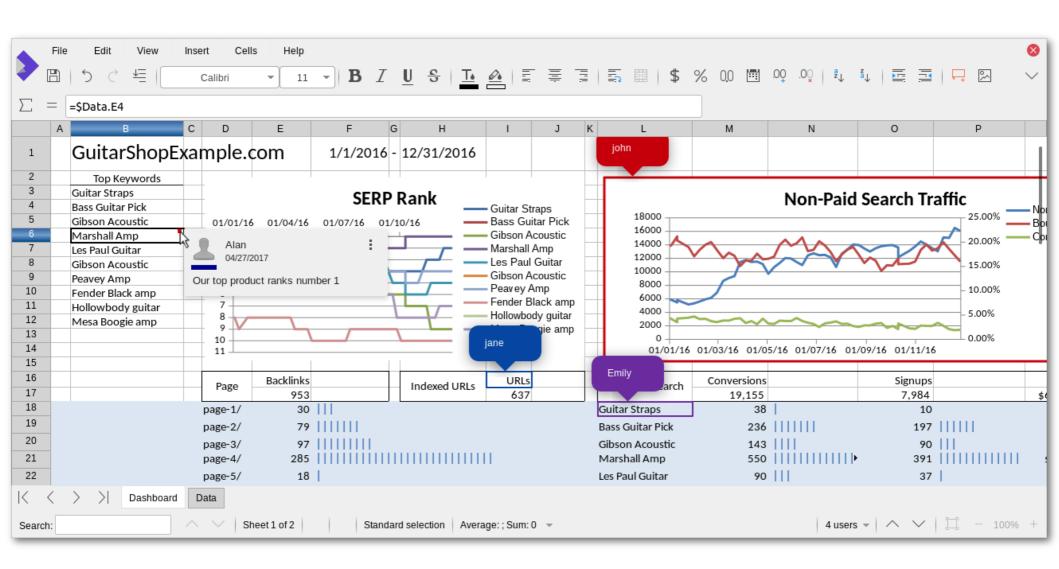
Online - moving to the browser

Richly featured collaborative editing ...



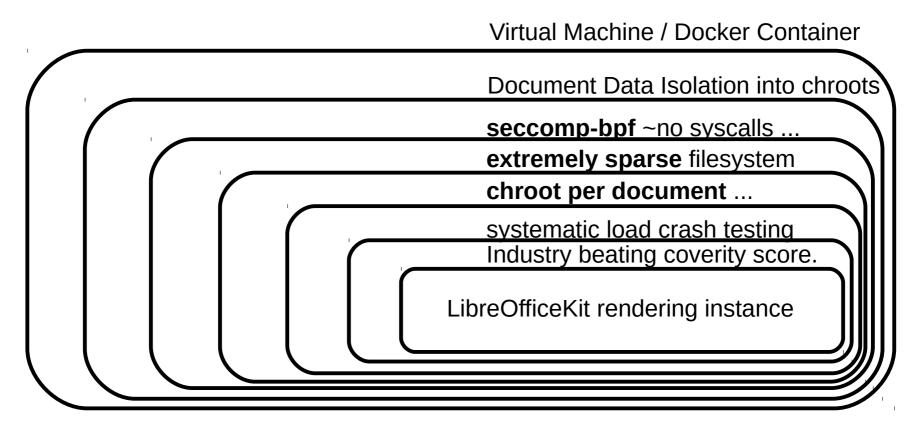
Online - moving to the browser

Richly featured collaborative editing ...



Online Design - The Onion

- Easy to deploy, integrates with lots of on-premise FLOSS EFSS
 - Nextcloud, ownCloud, pydio, seafile and lots more eg. Kolab



FLOSS / Security Methodologies ...

"First do ..."

- We are not at the 'before doing XYZ' stage
 - Everything we do is deep into the 'maintenance' box.
- New feature / function
 - Individuals work in fairly isolated areas to integrate their work.
 - **Agile:** "Each iteration involves a cross-functional team working in all functions: planning, analysis, design, coding, unit testing, and acceptance testing."

Re-factoring & architecting for security

- Significant scale / function re-work matter of man years.
- Permanent, ongoing incrementalism & mitigation.

Open Source volunteers

working code arrives - with no apparent methodology.

Auditing vs. Fuzzing ...

Auditing vs. Fuzzing vs. UI testing ...

Do they tackle different domains?

- Humans have intuitive skills, can focus on hot areas
- Humans are slow, imprecise, can propagate ~few assumptions through ~few stack-frames, and are expensive.
 - 1 million LOC added & subtracted each year is a lot
 - 1 full-time auditor's worth at least.

Probably both are required for now

- But ... the Al's are good, and are getting much better.
- Tending the automation is real work though ... as is connecting it up.
 - Adding targets to hunt also vital;
 - eg. if I break openSSL how do I know I 'got in'
 - → needs explicit instrumentation

Links / Further reading.

- Coverity Scan: LibreOffice
 - https://scan.coverity.com/projects/211
- LibreOffice & Online
 - Crash Testing results http://dev-builds.libreoffice.org/crashtest/?C=M&O=D
 - Online download https://www.collaboraoffice.com/code/
- OSS-Fuzz Announced
 - https://testing.googleblog.com/2016/12/announcing-oss-fuzz-continuous-fuzzing.html
- OSS-Fuzz Results (all reproducible fixed)
 - https://bugs.chromium.org/p/oss-fuzz/issues/list?can=1&q=libreoffice
- American Fuzzy Lop
 - https://en.wikipedia.org/wiki/American_fuzzy_lop_(fuzzer)
- Clang / Address Sanitizer / UbiSan
 - https://clang.llvm.org/docs/AddressSanitizer.html
 - https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html

Conclusions: Document security is tough.

- Open Source etc.
 - but security overseen by RedHat & other enterprises
- A flow process
 - harmed by regular mis-use of CVE process
 - Active mitigation & improvement work constantly ongoing
- Auditing alone is a waste of time & money
 - Unless heavily assisted by automation & integrated into your development flow – QA also susceptible to computation ...
- Tests running continuously: as you read this.
- Thank you for supporting LibreOffice!

Oh, that my words were recorded, that they were written on a scroll, that they were inscribed with an iron tool on lead, or engraved in rock for ever! I know that my Redeemer lives, and that in the end he will stand upon the earth. And though this body has been destroyed yet in my flesh I will see God, I myself will see him, with my own eyes - I and not another. How my heart yearns within me. - Job 19: 23-27