Relying on FOSS - Risk Perspectives

FOSSCOMM 2016

Dimitris Glynos

dimitris *at* census-labs.com
@dfunc on Twitter

The importance of FOSS

- From CIO.gov

  ... using and contributing back to open source software **can fuel innovation, lower costs, and benefit the public**.

- Gartner, 2015
  - 1.3 billion devices run Linux-based Android
- Jacob Appelbaum, #31C3
  - OTR and GnuPG seem to have evaded state-sponsored eavesdropping :-)

The Cathedral & the Bazaar

- Two very different models of development
- We would like to think that both build software for a purpose
- Cathedral
  - Software built by an **organization**
  - Closely follows and supports the customer demands
- Bazaar
  - Software built by the **community**
  - Features are built and maintained based on the needs (and views) of the **community**

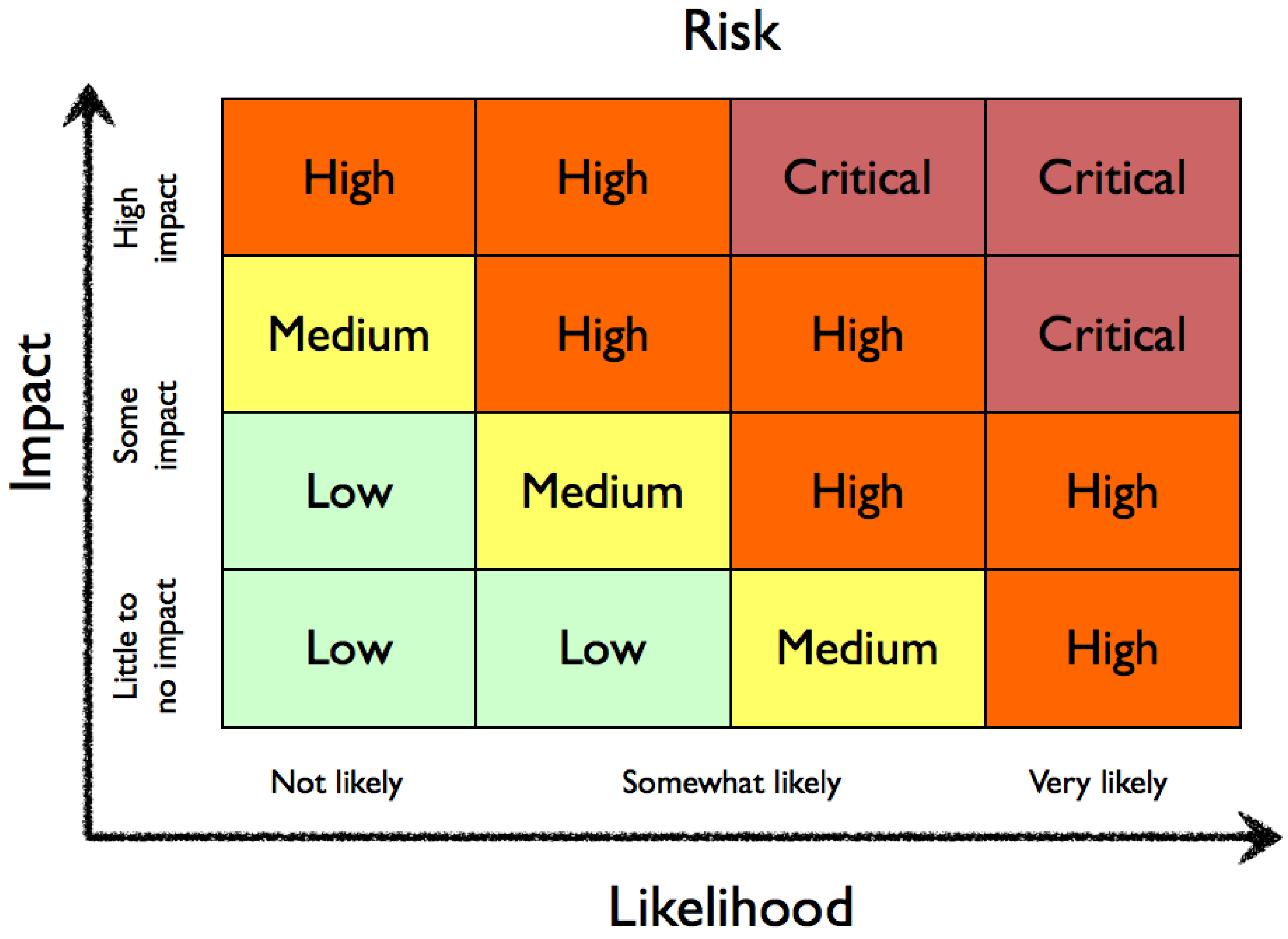
Theme of this talk

- Risk perspectives related to the use / creation / maintenance of FOSS within an organization
- Please note
  - we will not be considering risks related to the *adoption* of FOSS (e.g. hidden costs of ownership)

Definition of Risk

- Risk is the Likelihood of Danger
  - Risk = Likelihood x Impact
- Organizations **identify** and **measure** risks in order to better **handle** and **mitigate** them
  - Business Impact Analysis
  - Risk Assessment
  - Security Assessments
  - ...

Not all risks are the same



Risk

Impact	Not likely		Somewhat likely	Very likely
High impact	High	High	Critical	Critical
Some impact	Medium	High	High	Critical
	Low	Medium	High	High
Little to no impact	Low	Low	Medium	High

Likelihood

Exposure to FOSS

- Actual risk depends on the type and extent of exposure
  - Do you use FOSS to support internal processes and to what extent?
  - Do you use FOSS to develop software?
  - Do you use FOSS in a service you provide?
  - Do you use FOSS in a product you provide?
  - Do you maintain your own FOSS project?
  - Do you contribute to FOSS projects?

Risk #0: The invisible asset

- FOSS software is sometimes not accounted for in an IT infrastructure
  - Taken for granted; will the project be there tomorrow?
  - Not accounted for during risk assessment
  - Sometimes security updates and other bugfixes are not applied

Risk #0: The invisible asset

- Proposals for administrators
  - Record all distributions and major FOSS components used in the infrastructure
  - Refrain from using custom builds
  - Make sure all security (and other) policies apply to FOSS components as well
  - Provide dev teams with usage information

Risk #1: Maintenance

- Your FOSS-ninja (read: highly-skilled administrator / developer etc.) decides to leave the company
- Will you be able to find a substitute easily?
- Is the transition period going to be short and smooth?

Risk #1: Maintenance

- Proposals / Notes
  - As a society we must provide more opportunities for education in FOSS topics
  - My guess: there will always be room for subcontracting FOSS work
  - FOSS allows for great (and unmaintainable) patchwork; as a community we must adopt best practices for building maintainable systems

Risk #2: Customization

- You have to customize a certain software in order to fulfill your needs
  - Requires skill
  - Requires time
  - Requires maintenance of out-of-tree-patches

Risk #2: Customization

- Proposals
  - Organizations must make the effort to contribute (and maintain) their patches upstream. They will be benefiting in the same way, from contributions made from others.
  - Having someone on-board with the ability to customize software may be costly but is also an investment.
  - Individual users and organizations should engage more closely with FOSS dev. teams, voicing their concerns about missing functionality in projects.

Risk #3: Change

- **Critical** change in project
  - Users lose desired / needed functionality
  - May need to look for substitute project
- **Frequent** change in project
  - Project becomes at times unusable
  - May seriously affect provided services

Risk #3: Change

- Proposals
  - Be counted for! The project needs to know that you're using a specific functionality
  - If this project is important for you, engage more closely (join mailing lists, follow conferences etc.)
  - Organizations that depend on certain functionality should fund the development and maintenance of a 'stable' branch

Risk #4: Compatibility and Interoperability

- You may find that the software you use is not compatible / interoperable with other software or devices
- Very common with new hardware

Risk #4: Compatibility and Interoperability

- Proposals
  - Administrators may take a preference to vendors providing compatibility / interoperability drivers and middleware
  - Voice your concerns to the FOSS project
  - Voice your concerns to the vendor
  - If it's that important, fund it

Risk #5: Quality

- A “hacky” codebase with no documentation
- A codebase containing many security defects
- Code that sometimes does not work
  - Remember that “NO WARRANTIES” phrase in the LICENSE file?
- Code maturity is not easy to achieve
  - It requires an ongoing process that may not be feasible in a poorly funded FOSS project
  - Remember the OpenSSL Heartbleed bug?

Risk #5: Quality

- Proposals
  - If you feel the code/docs are a mess, help fix it.
  - Organizations that adopt FOSS must take the burden to properly audit the software (and contribute the findings of course)
  - Aside from the above, developers may also use automated tools to perform build, functional and security testing

Risk #6: Responsiveness

- How fast does the project team respond to:
  - a security bug disclosure?
  - a feature request?
  - an email?
- Slow response times are usually signs that a project is undermanned
- Does the project have a grumpy lead dev? :-)

Risk #6: Responsiveness

- Proposals
  - Organizations that depend on the responsiveness of a project team should donate time and money to the project
  - Development teams should be (more) welcoming to younger crowds that may have more time available. GSoCs are a great way to start.
  - If you find you can't work with a certain team there may be a similar project where your contributions will be of value.

Risk #7: Project dies

- The project is no longer maintained
- The project is no longer part of a software distribution
- The documentation site is lost
  - Remember Gentoo docs?

Risk #7: Project dies

- Proposals
  - Investigate (proactively) for alternatives
  - Step up to maintain
  - Summon other interested parties to resurrect it
  - Learn useful lessons from the dead project's history

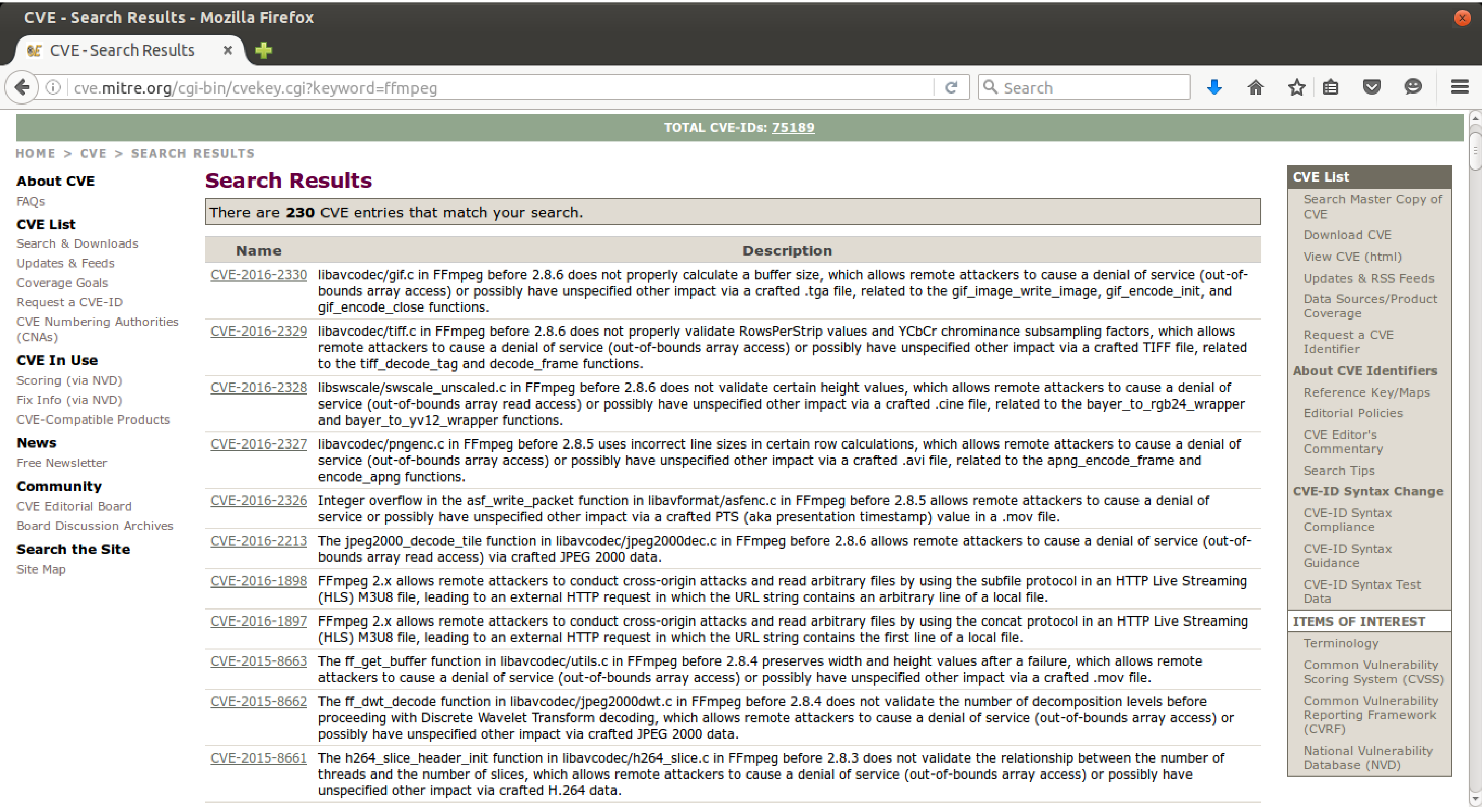
Risk #8: Forks

- Forks are too easy
- Forks create complexity
  - Imagine keeping track of important bugs on two or more projects
- Forks divide the workforce
- Forks create empathies in the community
- Forks are sometimes the only way
  - Anyone remember cdrecord?

The ffmpeg story (part 1)

- ffmpeg is an LGPL native library for media processing
- Bugs in ffmpeg may cause memory corruption
- Bugs in ffmpeg may under certain conditions allow for remote code execution
- See numerous Android stagefright bugs related to ffmpeg code

ffmpeg on cve.mitre.org (230 vulns)



CVE - Search Results - Mozilla Firefox

cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ffmpeg

TOTAL CVE-IDs: 75189

HOME > CVE > SEARCH RESULTS

About CVE
FAQs
CVE List
Search & Downloads
Updates & Feeds
Coverage Goals
Request a CVE-ID
CVE Numbering Authorities (CNAs)
CVE In Use
Scoring (via NVD)
Fix Info (via NVD)
CVE-Compatible Products
News
Free Newsletter
Community
CVE Editorial Board
Board Discussion Archives
Search the Site
Site Map

Search Results

There are **230** CVE entries that match your search.

Name	Description
CVE-2016-2330	libavcodec/gif.c in FFmpeg before 2.8.6 does not properly calculate a buffer size, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via a crafted .tga file, related to the gif_image_write_image, gif_encode_init, and gif_encode_close functions.
CVE-2016-2329	libavcodec/tiff.c in FFmpeg before 2.8.6 does not properly validate RowsPerStrip values and YCbCr chrominance subsampling factors, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via a crafted TIFF file, related to the tiff_decode_tag and decode_frame functions.
CVE-2016-2328	libswscale/swscale_unscaled.c in FFmpeg before 2.8.6 does not validate certain height values, which allows remote attackers to cause a denial of service (out-of-bounds array read access) or possibly have unspecified other impact via a crafted .cine file, related to the bayer_to_rgb24_wrapper and bayer_to_yv12_wrapper functions.
CVE-2016-2327	libavcodec/pngenc.c in FFmpeg before 2.8.5 uses incorrect line sizes in certain row calculations, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via a crafted .avi file, related to the apng_encode_frame and encode_apng functions.
CVE-2016-2326	Integer overflow in the asf_write_packet function in libavformat/asfenc.c in FFmpeg before 2.8.5 allows remote attackers to cause a denial of service or possibly have unspecified other impact via a crafted PTS (aka presentation timestamp) value in a .mov file.
CVE-2016-2213	The jpeg2000_decode_tile function in libavcodec/jpeg2000dec.c in FFmpeg before 2.8.6 allows remote attackers to cause a denial of service (out-of-bounds array read access) via crafted JPEG 2000 data.
CVE-2016-1898	FFmpeg 2.x allows remote attackers to conduct cross-origin attacks and read arbitrary files by using the subfile protocol in an HTTP Live Streaming (HLS) M3U8 file, leading to an external HTTP request in which the URL string contains an arbitrary line of a local file.
CVE-2016-1897	FFmpeg 2.x allows remote attackers to conduct cross-origin attacks and read arbitrary files by using the concat protocol in an HTTP Live Streaming (HLS) M3U8 file, leading to an external HTTP request in which the URL string contains the first line of a local file.
CVE-2015-8663	The ff_get_buffer function in libavcodec/utils.c in FFmpeg before 2.8.4 preserves width and height values after a failure, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via a crafted .mov file.
CVE-2015-8662	The ff_dwt_decode function in libavcodec/jpeg2000dwt.c in FFmpeg before 2.8.4 does not validate the number of decomposition levels before proceeding with Discrete Wavelet Transform decoding, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via crafted JPEG 2000 data.
CVE-2015-8661	The h264_slice_header_init function in libavcodec/h264_slice.c in FFmpeg before 2.8.3 does not validate the relationship between the number of threads and the number of slices, which allows remote attackers to cause a denial of service (out-of-bounds array access) or possibly have unspecified other impact via crafted H.264 data.

CVE List
Search Master Copy of CVE
Download CVE
View CVE (html)
Updates & RSS Feeds
Data Sources/Product Coverage
Request a CVE Identifier
About CVE Identifiers
Reference Key/Maps
Editorial Policies
CVE Editor's Commentary
Search Tips
CVE-ID Syntax Change
CVE-ID Syntax Compliance
CVE-ID Syntax Guidance
CVE-ID Syntax Test Data
ITEMS OF INTEREST
Terminology
Common Vulnerability Scoring System (CVSS)
Common Vulnerability Reporting Framework (CVRF)
National Vulnerability Database (NVD)

The ffmpeg story (part 2)

- ffmpeg is pretty important. It runs on
  - Your computer (browsers, vlc etc.)
  - Your mobile phone
  - Your streaming media box
  - Your TV
  - In infotainment systems of cars
  - In infotainment systems of airplanes
- And is also <u>forked</u> (remember libav?)
  - Lead dev resigned over this in August 2015

Risk #8: Forks

- Proposal
  - Don't create unnecessary forks
  - Don't support unnecessary forks
  - Spend the time to contribute to the existing project
  - Have face-2-face meetings with the dev team to explain your views
  - Consider forks as projects that have a significantly different goal
    - Ideally make shared code a library. Don't embed code “as is”.

Risk #9: Code Integrity

- Malicious injection of code in the project or project bundles
  - Remember the OpenBSD backdoor ?
  - Remember the ProFTPD backdoor ?

Risk #9: Code Integrity

- Proposals
  - Development teams should take every measure possible to minimize this risk.
  - Organizations must audit the software they use and its related infrastructure. Period.
  - The community must respond rapidly to such threats.
  - Signatures from devs help.
  - Reproducible builds help also.

Risk #10: © Infringement

- Contributing to a project that gets a copyright infringement letter
- Are you protected?

Risk #10: © Infringement

- Proposals
  - There's some propaganda out there that scares org's from contributing to FOSS. Seriously, don't worry that much about it.
  - Before you commit code, check if it is suitable for incorporation to the project and compatible with the project's license
  - Seek advice
    - Software Freedom Law Center
    - FSF Compliance Lab Team
    - European Legal Network – FSF Europe
    - Linux Foundation Legal Defense Fund
    - ...

Conclusions

- Are we ready for world domination?
- Sustainable FOSS requires an active and engaging user base.
- Quality FOSS requires similar processes and funds as those available to proprietary software. Organizations must help in this regard.

Useful references

- Eric S. Raymond, “The Cathedral and the Bazaar”, ISBN 1-56592-724-9.
- Federal Financial Institutions Examination Council, *“Risk Management of Free and Open Source Software*”, available at http://www.federalreserve.gov/boarddocs/srletters/2004/sr0417a1.pdf
- CVE – Common Vulnerabilities and Exposures, https://cve.mitre.org
- OSS-security mailing list, http://www.openwall.com/lists/oss-security/
- Linux Foundation Legal Program

  http://www.linuxfoundation.org/programs/legal
- FSF Europe Legal Network

  https://fsfe.org/activities/ftf/network.en.html

Questions?