Open Source Software (OSS)

David A. Wheeler

March 12, 2007

This presentation contains the views of the author and does not indicate endorsement by IDA, the U.S. government, or the U.S. Department of Defense.

Outline

- Introduction to OSS
  - What is it?
  - Nearly all OSS is commercial off-the-shelf (COTS)
  - Typical OSS development model
- Value to government
  - Why would governments use or create OSS?
  - OSS Challenges
  - Examples of use
- Selecting COTS OSS: What’s the Same? Different?
- Starting OSS project
- Security
- Foolish vs. Sensible Policies
- Remarks

What is Open Source Software (OSS)?

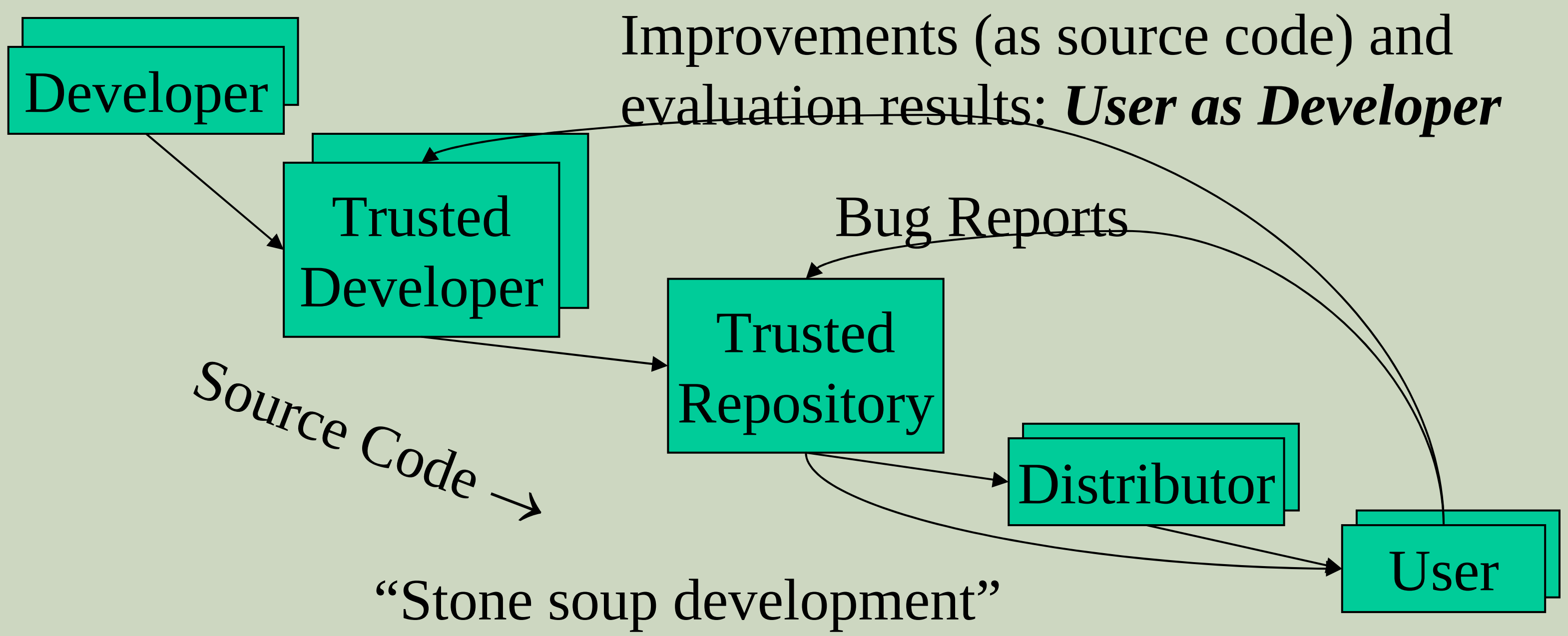
- OSS: software licensed to users with these freedoms:
  - to run the program for any purpose,
  - to study and modify the program, and
  - to freely redistribute copies of either the original or modified program (without royalties, etc.)
- Synonyms: libre software, Free software*, FOSS, FLOSS
- Antonyms: proprietary software, closed software
- Widely used; OSS #1 or #2 in many markets
  - "… plays a more critical role in the DoD than has generally been recognized." [MITRE 2003]
- <u>*Not* non-commercial…</u>

* The term "Free software" sometimes means OSS, and sometimes instead means "no charge"

Nearly all OSS is Commercial Off-the-Shelf (COTS)

- **Federal Acquisition Regulation (FAR) prefers COTS and NDI; commercial item = "licensed to general public":**
  - Agencies must "(a) Conduct market research to determine [if] commercial items or nondevelopmental items are available … (b) Acquire [them] when… available … (c) Require prime contractors and subcontractors at all tiers to incorporate, to the maximum extent practicable, [them] as components..."
  - Commercial item is "(1) Any item, other than real property, that is of a type customarily used by the general public or by non-governmental entities for purposes [not unique to a government], and (i) Has been <u>sold</u>, leased, or <u>licensed</u> to the general public; or (ii) Has been <u>offered</u> for <u>sale</u>, lease, or <u>license</u> to the general public... (3) [Above with] (i) Modifications of a type customarily available in the commercial marketplace; or (ii) Minor modifications… made to meet Federal Government requirements. "
  - True for nearly all off-the-shelf (OTS) OSS, so it's commercial item/COTS
- **OSS projects usually seek improvements = financial gain**
  - U.S. Code Title 17, section 101 defines "financial gain" as including "receipt, or expectation of receipt, of anything of value, <u>including the receipt of other copyrighted works</u>."
- **Many OSS projects supported by commercial companies**
  - IBM, Sun, Red Hat, Novell, Microsoft (WiX, IronPython, SFU, Codeplex site)
- **Often developers paid (2004: 37K/38K Linux changes)**
- **OSS licenses and projects approve of commercial support**
- **Use COTS/NDI because *<u>users share costs</u>* – OSS does!**

OSS Development Model



- OSS users typically use software without paying licensing fees
- OSS users typically pay for training & support (competed)
- OSS users are responsible for developing new improvements & any evaluations that they need; often cooperate/pay others to do so

Why would governments use or create OSS (value for government)?

- Can evaluate in detail, lowering risk
  - Can see if meets needs (security, etc.)
  - Mass peer review typically greatly increases quality/security
  - Aids longevity of records, government transparency
- Can copy repeatedly at no additional charge (lower TCO)
  - Support may have per-use charges (compete-able)
- Can share development costs with other users
- Can modify for special needs & to counter attack
  - Even if you're the only one who needs the modification
- *Control own destiny*: Freedom from vendor lock-in, vendor abandonment, conflicting vendor goals, etc.

In many cases, OSS approaches have the *potential* to increase functionality, quality, and flexibility, while lowering cost and development time

OSS Challenges

1. Ensuring OSS fairly considered in acquisitions
   - Some acquisition processes/policies not updated for OSS
   - Policy noncompliance (FAR's market research, "OSS neutral")
   - Many PMs unfamiliar with OSS: don't consider using or creating
   - Many COTS OSS projects ignore solicitations & RFPs
2. Different economics: Pay-up-front for improvements
   - Some policies presume proprietary COTS' pay-per-use model
   - Can pay in $ or time, can compete, can cost-share with other users
3. Transition costs if pre-existing system
   - Especially if dependent on proprietary formats/protocols/APIs
   - Use open standards so can switch (multi-vendor, no 'RAND' patents)
   - Web-based apps/SOA help if open stds (browser/platform-neutral)
   - Vendor lock-in often increases TCO; transition may be worthwhile
4. COTS support if no traditional vendor (compete-able)
5. License compliance (easier but different: education)
6. Cannot release classified code as OSS
   - Can build classified systems with/incl. OSS: tables, layers, licenses

Examples of OSS in U.S. Government

- Use – ***pervasive***
  - OSS "plays a more critical role in the DoD than has generally been recognized"; inc. Linux, Samba, Apache, Perl, GCC, GNAT, XFree86, OpenSSH, bind, and sendmail. [MITRE 2003]
  - "devIS saves its clients a minimum of $100,000 per contract by using OSS" [NewsForge]
  - Often unaware it's OSS
- Government-paid improvements of OSS
  - OpenSSL (CC evaluation), Bind (DNSSEC), GNAT, …
- Government-developed OSS
  - BSD TCP/IP suite, Security-Enhanced Linux (SELinux), OpenVista, Expect, EZRO, Evergreen (Georgia), …
- U.S. federal policies explicitly neutral: OSS, or not, is fine
  - OMB memo M-04-16, DoD memo "OSS in DoD"
  - Examine *all* licenses before commit (GPL fine)

Selecting COTS: What’s the Same? (OSS vs. Proprietary)

- Negotiate best options with all parties, *then* select
- Evaluate by winnowing out top candidates for your needs
  - *Identify* candidates, Read *Reviews, Compare* (briefly) to needs through criteria, *Analyze* top candidates
- Evaluation criteria - same
  - Functionality, total cost of ownership, support, maintenance/ longevity, reliability, performance, scalability, flexibility, legal/license (inc. *rights and responsibilities – OSS always gives right to view, modify, and redistribute*), mkt share, other
- Warranty & indemnification (“who do you sue?”)
  - Generally disclaimed by *both* proprietary & OSS licenses
- Pay for installation, training, support (time and/or money)
- Developer trustworthiness usually unknown
  - Mitigation: Can review OSS code & sometimes proprietary
  - Mitigation: Supplier due diligence; often main OSS developers and integrators determinable
  - Remember: Selling company often not developer

Selecting COTS: What’s Different? (OSS vs. Proprietary)

- Process/code openness means more&different sources of evaluation information for COTS OSS
  - Bug databases, mailing list discussions, …
  - Anyone (inc. you) can evaluate in detail
  - See http: //www.dwheeler.com/oss_fs_eval.html
- Proprietary=pay/use, OSS=pay/improvement
  - In OSS, pay can be time and/or money
- Support can be competed & changed
  - OSS vendors, government support contracts, self
- OSS can be modified & redistributed
  - New option, but need to know when to modify
  - Forking usually fails; generally work with community

Starting OSS Project

- Check usual project-start requirements
  - Is there a need, no/better solution, TCO, etc.
  - Examine OSS approach; similar to GOTS, with greater opportunity for cost-sharing, but greater openness
- Purpose is *cost-sharing*: remove barriers to entry
  - Use common license well-known to be OSS (GPL, LGPL, MIT/X, BSD-new) – *don't write your own license*
  - Establish project website (mailing list, tracker, source)
  - Document scope, major decisions
  - Use typical infrastructure, tools, etc. (e.g., SCM)
  - Maximize portability, avoid proprietary langs/libraries
  - *Must run* - Small-but-running better than big-and-not
  - Establish vetting process(es) before government use
    - Government-paid lead? Testing? Same issues: proprietary
- Many articles & books on subject

Security

- Neither OSS nor proprietary are *always* more secure
  - Many *specific* OSS programs *are* significantly more secure; see quantitative studies “Why…” at http://www.dwheeler.com
- OSS advantage: Open design principle
  - Saltzer & Schroeder [1974/1975], “Protection mechanism must not depend on attacker ignorance”
- Hiding source code doesn’t impede attacks
  - “Security by Obscurity” requires *real* secret-keeping: can’t give access to source code, executable program, or website
- Attackers can modify OSS *and* proprietary software
  - Trick is to get that modified version into supply chain
  - OSS: subverting/misleading/becoming the trusted developers or trusted repository/distribution, *and* none notice attack later
- OSS security requirements:
  - Developers/reviewers need security knowledge
  - People have to actually review the code: yes, it really happens!
  - Problems must be fixed, fixes deployed

Foolish vs. Sensible Policies

- **Foolish: “No OSS” or “No GPL”**
  - ***Tremendous*** **competitive/strategic disadvantage**
  - **Essentially the same idea as “no COTS” decades ago**
- **Often focused on General Public License (GPL):**
  - **~ “Someone given binary *must* get source code too”**
  - **GPL is most popular OSS license *by far* (52%-88%)**
  - **Some proprietary companies advocate “no GPL” as veiled anti-OSS campaign, to inhibit competition**
- **Sensible: “Examine OSS & proprietary options, then review *all* their licenses before including”**
  - **Ensure all licenses are compatible with intended use**
  - **Proprietary EULAs sometimes *worse* than OSS licenses**
  - **GPL often fine once considered in context**
  - ***Examine supplier*** **– again, for OSS *and* proprietary**

Concluding Remarks

- **OSS options should always be considered**
  - **Both choosing COTS OSS & creating new OSS project**
  - **Components or even whole project (depending on need)**
  - **Not always best choice, but foolish to ignore**
- **OSS can be very flexible & often lowers costs**
  - **Directly and as competition to non-OSS (keep options open!)**
- **OSS raises strategic questions for governments**
  - **How pool users to start OSS projects when appropriate?**
  - **Educating PMs on OSS, deploying fully open architectures**
  - **Research: default to OSS (with some common OSS license)**
  - **Eliminating software patents**
- **Projects should change to consider OSS approaches:**
  - **PM education: OSS differences, fears, *always consider option***
  - **Classified systems: separate data & program, layer programs**
  - **Open standards so can change later (e.g., browser-neutral)**
    - ***Require & operationally demonstrate* that can switch components**

Security Backup Slides

Extreme claims

- Extreme claims
  - “FLOSS is always more secure”
  - “Proprietary is always more secure”
- Reality: Neither FLOSS nor proprietary always better
  - Some *specific* FLOSS programs *are* more secure than their competing proprietary competitors
- Include FLOSS options when acquiring, then evaluate

FLOSS Security (1)

- **Browser "unsafe" days in 2004: 98% Internet Explorer, 15% Mozilla/Firefox (half of Firefox's MacOS-only)**
- **IE 21x more likely to get spyware than Firefox [U of Wash.]**
- **Faster response: Firefox 37 days, Windows 134.5 days**
- **Evans Data: Linux rarely broken, ~virus/Trojan-free**
- **Serious vulnerabilities: Apache 0, IIS 8 / 3yrs**
- **J.S. Wurzler hacker insurance costs 5-15% more for Windows than for Unix or Linux**
- **Bugtraq vulnerability 99-00: Smallest is OpenBSD, Windows largest (**Don't quintuple-count!**)**
- **Windows websites more vulnerable in practice**

Category	Proprietary	FLOSS
Defaced	66% (Windows)	17% (GNU/Linux)
Deployed Systems	49.6% (Windows)	29.6% (GNU/Linux)
Deployed websites (by name)	24.81% (IIS)	66.75% (Apache)

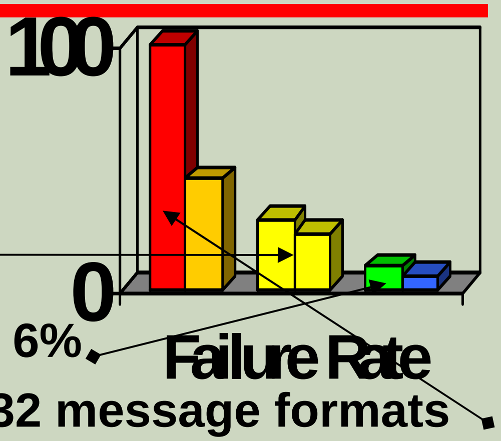
FLOSS Security (2)

- Unpatched networked systems: 3 months Linux, hours Windows (variance minutes ... months) [Honeynet.org, Dec 2004]
  - Windows SP2 believed to be better than previous versions of Windows
- 50% Windows vulnerabilities are critical, vs. 10% in Red Hat [Nicholas Petreley, Oct 2004]
- Viruses primarily Windows phenomenon
  - 60,000 Windows, 40 Macintosh, 5 for commercial Unix versions, 40 for Linux
- 91% broadband users have spyware on their home computers (proprietary OS) [National Cyber Security Alliance, May 2003] vs. ~0% on FLOSS

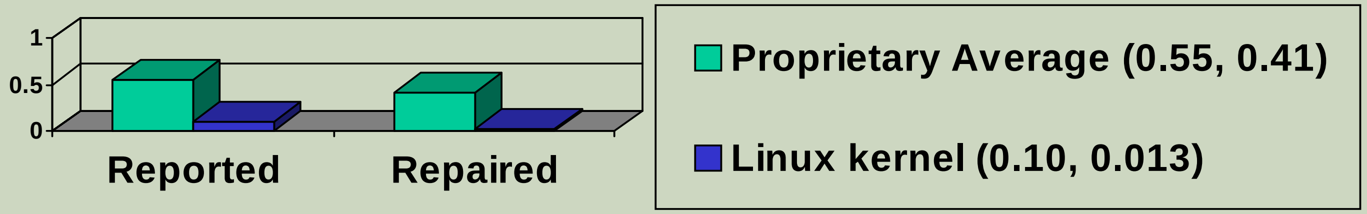
FLOSS Security (3)

- FLOSS systems scored better on security [Payne, Information Systems Journal 2002]
- Survey of 6,344 software development managers April 2005 favored FLOSS [BZ Research]

Reliability



- Fuzz studies found FLOSS apps significantly more reliable [U Wisconsin]
  - Proprietary Unix failure rate: 28%,23%
  - FLOSS: Slackware Linux 9%, GNU utilities 6%
  - Windows: 100%; 45% if forbid certain Win32 message formats
- GNU/Linux vs. Windows NT 10 mo study [ZDNet]
  - NT crashed every 6 weeks; both GNU/Linuxes, never
- IIS web servers >2x downtime of Apache [Syscontrol AG]
- Linux kernel TCP/IP had smaller defect density [Reasoning]



FLOSS Always More Secure?

- **No: Sendmail, bind 4**
- **Must examine case-by-case**
  - **But there *is* a principle that gives FLOSS a *potential* advantage…**

Open design: A security fundamental

- Saltzer & Schroeder [1974/1975] - Open design principle
  - the protection mechanism must not depend on attacker ignorance
- FLOSS better fulfills this principle
- Security experts perceive FLOSS advantage
  - Bruce Schneier: “demand OSS for anything related to security”
  - Vincent Rijmen (AES): “forces people to write more clear code & adhere to standards”
  - Whitfield Diffie: “it’s simply unrealistic to depend on secrecy for security”

Problems with hiding source & vulnerability secrecy

- Hiding source doesn't halt attacks
  - Presumes you can keep source secret
    - Attackers may extract or legitimately get it
  - Dynamic attacks don't need source or binary
    - Observing output from inputs sufficient for attack
  - Static attacks can use pattern-matches against binaries
  - Source can be regenerated by disassemblers & decompilers sufficiently to search for vulnerabilities
  - Secrecy inhibits helpers, while not preventing attackers
  - "Security by Obscurity" widely denegrated
- Hiding source slows vulnerability response
- Vulnerability secrecy doesn't halt attacks
  - Vulnerabilities are a time bomb and are likely to be rediscovered by attackers
  - Brief secrecy works (10-30 days), not months/years

Can “Security by Obscurity” be a basis for security?

- “Security by Obscurity” can work, but iff:
  - Keeping secret actually improves security
  - You can keep the critical information a secret
- For obscurity itself to give significant security:
  - Keep source secret from all but a few people. Never sell or reveal source to many. E.G.: Classify
  - Keep binary secret; never sell binary to outsiders
    - Use software protection mechanisms (goo, etc.)
    - Remove software binary before exporting system
  - Do not allow inputs/outputs of program to be accessible by others – *no Internet/web access*
- ***<u>Useless</u>* in most cases!**
  - Incompatible with proprietary off-the-shelf model
- Proprietary software can be secure – but not this way

FLOSS Security Preconditions (Unintentional vulnerabilities)

1. Developers/reviewers need security knowledge
   - Knowledge more important than licensing
2. People have to actually *review* the code
   - Reduced likelihood if niche/rarely-used, few developers, rare computer language, not really FLOSS
   - More contributors, more review
     - Is it *truly community-developed?*
   - Evidence suggests this really happens! (next)
3. Problems must be fixed
   - Far better to fix *before* deployment
   - If already deployed, need to deploy fix

Is FLOSS code ever examined? Yes.

- Most major FLOSS projects have specific code reviews; some have rewards
  - Mozilla Security Bug Bounty Program ($500)
  - Linux: hierarchical review, “sparse” tool
- Disseminated review groups (second check):
  - OpenBSD (for OpenBSD)
  - Debian-audit (for Debian Linux)
- Static analysis tool vendors test using FLOSS
- Vulnerability Discovery and Remediation, Open Source Hardening Project (DHS/Coverity/Stanford)
- Many independents (see Bugtraq, etc.)
- Business case: Must examine to change (*reason to review)*
- Users' increased transparency encourages examination & feedback

Evaluating FLOSS? Look for evidence

- First, identify your security requirements
- Look for evidence at FLOSS project website
  - User's/Admin Guides: discuss make/keep it secure?
  - Process for reporting security vulnerabilities?
  - Cryptographic signatures for current release?
  - Developer mailing lists discuss security issues and work to keep the program secure?
  - Active community
- Use other information sources where available
  - E.G., CVE… but absence is not necessarily good
  - External reputation (e.g., OpenBSD)
- See http://www.dwheeler.com/oss_fs_eval.html

Proprietary advantages… not necessarily

- Experienced developers who understand security produce better results
  - Experience & knowledge *are critical*, but...
  - FLOSS developers often very experienced & knowledgeable too (BCG study: average 11yrs experience, 30 yrs old) – often same people
- Proprietary developers higher quality?
  - Dubious; FLOSS often higher reliability,security
  - Market rush often impairs proprietary quality
- No guarantee FLOSS is widely reviewed
  - ***True!*** Unreviewed FLOSS may be very insecure
  - Also true for proprietary (rarely reviewed!). *Check it!*
- Can sue vendor if insecure/inadequate
  - Nonsense. EULAs forbid, courts rarely accept, costly to sue with improbable results, you want sw not a suit

Inserting malicious code & FLOSS: Basic concepts

- “Anyone can modify FLOSS, including attackers”
  - Actually, you can modify proprietary programs too… just use a hex editor. Legal niceties not protection!
  - Trick is to get result into user supply chain
  - In FLOSS, requires subverting/misleading the trusted developers or trusted repository/distribution…
  - *and* no one noticing the public malsource later
- Different threat types: Individual...nation-state
- Distributed source aids detection
- Large community-based FLOSS projects tend to have many reviewers from many countries
  - Makes attacks more difficult
  - Consider supplier as you would proprietary software
  - Risk larger for small FLOSS projects

Malicious attack approaches: FLOSS vs. proprietary

- **Repository/distribution system attack**
  - **Traditional proprietary advantage: can more easily disconnect repository from Internet, not shared between different groups**
    - **But development going global, so disconnect less practical**
  - **Proprietary advantage: distribution control**
  - **OSS advantage: Easier detection & recovery via many copies**
- **Malicious trusted developers**
  - **OSS slight advantage via review, but weak ("fix" worse!)**
  - **OSS slight advantage: More likely to know who developers are**
  - **Reality: For both, *check who is developing it!***
- **Malicious untrusted developer**
  - **Proprietary advantage: Fewer untrusted developers**
    - **Sub-suppliers, "Trusted" developers may be malicious**
  - **OSS long-term advantages: Multiple reviewers (more better)**
- **Unclear winner – No evidence proprietary always better**

Examples: Malicious code & FLOSS

- Linux kernel attack – repository insertion
  - Tried to hide; = instead of ==
  - Attack failed (CM, developer review, conventions)
- Debian/SourceForge repository subversions
  - Countered & restored by external copy comparisons
- Often malicious code made to look like unintentional code
  - Techniques to counter unintentional still apply
  - Attacker could devise to work around tools... but won't know in FLOSS what tools are used!
- Borland InterBase/Firebird Back Door
  - user: politically, password: correct
  - Hidden for 7 years in proprietary product
  - Found after release as FLOSS in 5 months
  - Unclear if malicious, but has its form

Security Preconditions (Malicious vulnerabilities)

- Counter Repository/distribution system attack
  - Widespread copies, comparison process
  - Evidence of hardened repository
  - Digitally signed distribution
- Counter Malicious trusted developers
  - Find out who's developing your system (*always!*)
- Counter Malicious untrusted developer
  - Strong review process
    - As with unintentional vulnerabilities: Security-knowledgeable developers, review, fix what's found
  - Update process, for when vulnerabilities found

High Assurance

- High assurance (HA) software:
  - Has an argument that could convince skeptical parties that the software will *always perform or never perform* certain key functions *without fail*... convincing evidence that there are *absolutely* no software defects. CC EAL 6+
  - Significant use of formal methods, high test coverage
  - High cost – requires deep pockets at this time
  - A few OSS & proprietary tools to support HA dev
  - Few proprietary, even fewer OSS HA at this time
- Theoretically OSS should be better for HA
  - In mathematics, proofs are often wrong, so only peer review of proofs valid [De Millo,Lipton,Perlis]. OSS!
- HA developers/customers very conservative & results often secret, so rarely apply "new" approaches like OSS... yet
  - Cannot easily compare in practice... yet

Can FLOSS be applied to custom systems?

- **Effective FLOSS systems typically have built a large development community**
  - **Share costs/effort for development & review**
  - **Same reason that proprietary off-the-shelf works: Multiple customers distribute costs**
- **Custom systems can be built from FLOSS (& proprietary) components**
- **If no pre-existing system, sometimes can create a generalized custom system**
  - **Then *generalized* system FLOSS, with a custom configuration for your problem**
  - **Do risk/benefit analysis before proceeding**

Bottom Line

- Neither FLOSS nor proprietary always better
  - But clearly many cases where FLOSS *is* better
- FLOSS use increasing industry-wide
  - In some areas, e.g., web servers, it dominates
- Policies must not ignore or make it difficult to use FLOSS where applicable
  - Can be a challenge because of radically different assumptions & approach
- Include FLOSS options when acquiring, then evaluate

Quantitative Studies - Backup Slides

Outline of Quantitative Information on OSS/FS

Quantitatively show “consider using OSS/FS software”:

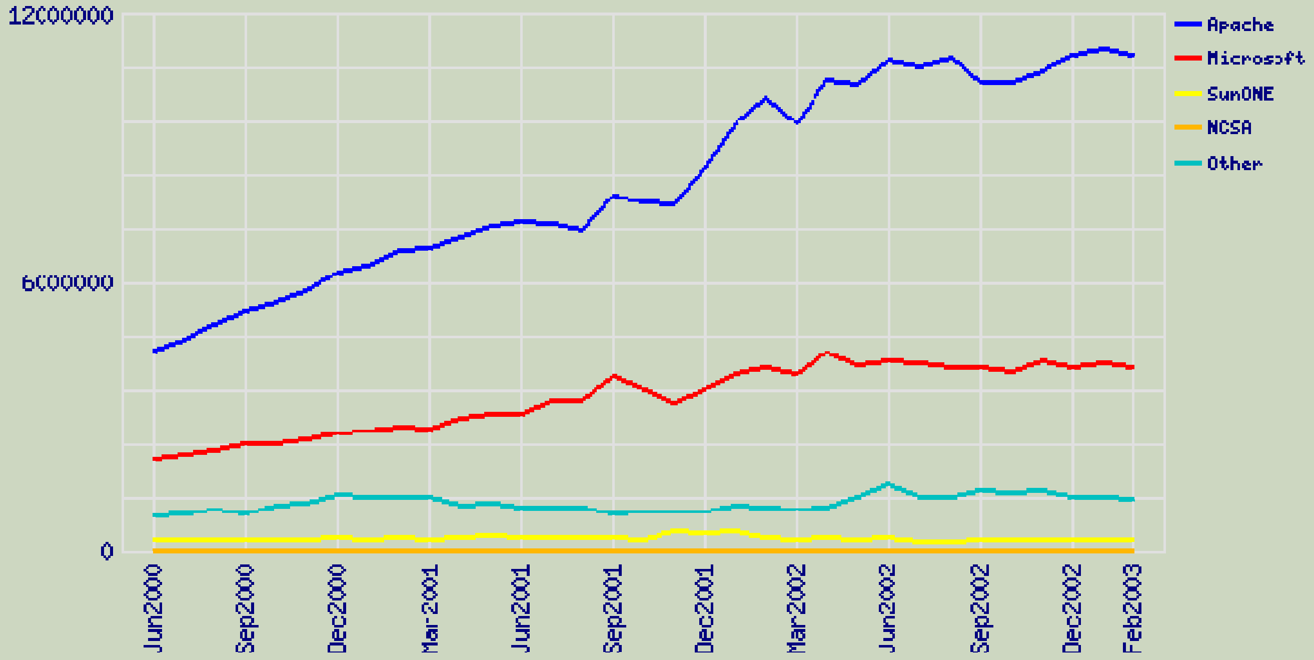
- Market Share
- Reliability
- Performance
- Scalability
- Security
- Total cost of ownership
- Non-quantitative

Numbers won't show OSS/FS always technically better

This presentation does not necessarily represent the views of the U.S. Government or U.S. DoD, & is based on personal work

Market Share: Web Servers

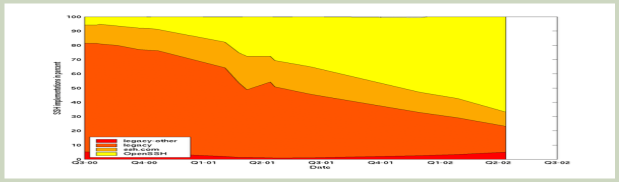
- **Active Sites: Apache 66.75%, Microsoft IIS 24.81% in Feb 2003 (counting by name; 35.86M sites)**



- **For SSL, Apache 53.97%, IIS 34.85% Sep02**

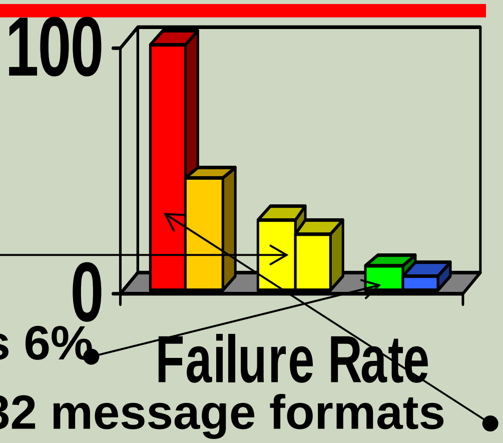
Other Market Share Examples

- GNU/Linux #2 webserver OS Jun01
  - GNU/Linux 29.6%, Windows 49.6%, BSDs 6.1%
- GNU/Linux #2 server OS sold 99, 00, 01 (24%, 27%, 25%)
- DNS: bind supports 95% of reverse-lookups
- PHP #1 server-side scripting language
- Sendmail #1 Email server
  - Sendmail 42%, Microsoft Exchange 18%
- OpenSSH #1 SSH (66.8% Apr02)

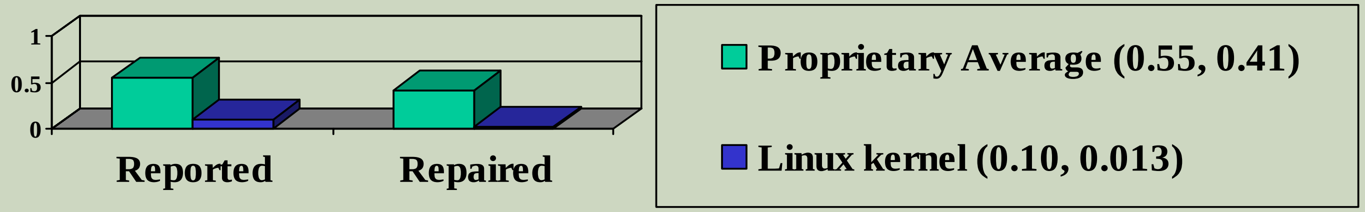


- Small (1.7-3.8%) 2002 desktop share
  - Microsoft 92% in 2000, but usable OSS/FS apps just released in 2002, so could change over time

Reliability



- Fuzz studies found OSS/FS apps significantly more reliable [U Wisconsin]
  - Proprietary Unix failure rate: 28%,23%
  - OSS/FS: Slackware Linux 9%, GNU utilities 6%
  - Windows: 100%; 45% if forbid certain Win32 message formats
- GNU/Linux vs. Windows NT 10 mo study [ZDNet]
  - NT crashed every 6 weeks; both GNU/Linuxes, never
- IIS web servers >2x downtime of Apache [Syscontrol AG]
- Linux kernel TCP/IP had smaller defect density [Reasoning]



Performance

- Performance always varies by circumstance
- TPC-C: GNU/Linux faster than Windows
- PC Magazine: GNU/Linux with Samba faster fileserving at Windows' own file protocols
  - Nov 2001, top end, 130MB/sec vs. 78MB/sec
  - April 2002, performance 2x; 4x many clients
- Sys Admin: untuned GNU/Linux fastest

Measure	GNU/Linux	Solaris on Intel	FreeBSD	Windows 2000
Email (M msg/hr)	**1.3**	**1**	**0.9**	**0.9**
Disk I/O (seconds)	**542**	**3990**	**2398**	**613**

Scalability

- GNU/Linux and NetBSD support more hardware platforms & performance ranges
  - PC hardware, PDAs, mainframes, clusters, supercomputers
- OSS/FS can develop large software systems
  - Red Hat Linux 7.1 had 30million SLOC
  - Represents approximately 8,000 person-years
  - To re-develop proprietary, $1 Billion USD

Security

- **J.S. Wurzler hacker insurance costs 5-15% more for Windows than for Unix or Linux**
- **Windows websites disproportionately vulnerable**

Category	Proprietary	OSS/FS
Defaced	**66% (Windows)**	**17% (GNU/Linux)**
Deployed Systems	**49.6% (Windows)**	**29.6% (GNU/Linux)**
Deployed websites (by name)	**24.81% (IIS)**	**66.75% (Apache)**

- **Bugtraq vulnerability: Smallest is OpenBSD, Windows largest (**Don't quintuple-count!**)**
- **Worst vulnerabilities (takeover): Apache 0, IIS 8 (Jun98-Jun01)**
- **OSS/FS *not invulnerable!***

Total Cost of Ownership (TCO)

- TCO multifaceted & sensitive to circumstances
- OSS/FS costs less to acquire than proprietary
  - E.G., Web server, Windows $3610 vs. $156
- Some other factors also tend to be lower
  - Lower upgrade costs, can use cheaper hardware
  - Avoids license management & litigation
- Cybersource: TCO 24%-34% less w/OSS/FS
- InfoWorld Survey of CTOs:
  - 60% CTOs: >$50K/yr savings
  - 32% CTOs: > $250K/yr savings (inc. above)

Non-Quantitative

- To many, non-quantitative advantages of OSS/FS are more important
  - Social/ethical/moral reasons
  - Avoids risks of single source solutions
    - Reversible decision: can switch or self-support if vendor jacks up price, maliciously changes interface, drops support, …
  - (Can) avoid security risks of monocultures
  - Avoids license management and litigation
  - Supports domestic IT infrastructure
  - Many believe it encourages innovation
  - Greater flexibility
    - Can change software (or hire its change) to meet needs

Conclusions on Quantitative OSS/FS Information

- Many, many cases where OSS/FS programs have some measurable advantage over proprietary competition
- Consider using OSS/FS software when acquiring software
- For more detailed information, see http://www.dwheeler.com/oss_fs_why.html

Conclusions

- Many similarities and differences in acquisition for OSS
  - Need to know & handle differences, challenges
- Need to know how to evaluate OSS
  - General approach similar
  - Ways to acquire information differ
- Quantitative evidence that OSS is worth considering
  - I don’t think it’s *always* the right answer, but it’s *always* worth considering

Miscellaneous Backup Slides

Major OSS/FS Licenses

- **Many licenses, but 4 dominate**
- **BSD-new & MIT license: anything but sue**
  - **Can incorporate code into proprietary software**
  - **Financial incentive to use, but *not* aid project**
- **General Public License (GPL): “Copyleft”**
  - **If distribute, *must* distribute source code or provide written offer to do so**
  - **Cannot link (embed) into proprietary software**
- **Lesser/Library GPL - a compromise**
  - **Must distribute source code/written offer, but only of component itself**
  - ***Can* link into proprietary software**
- **Public domain is OSS/FS, but rare**

GPL Use Widespread

- GPL has widespread use, other licenses far less common
  - Freshmeat.net (2003): 69.66% GPL, 5.29% LGPL, 4.82% BSD licenses (combined)
  - SourceForge.net (2003): 71% GPL, 10% LGPL, 7% BSD
  - Red Hat Linux 7.1: 50.36% GPL solely (55.3% dual), 8.28% MIT, 7.64% LGPL
  - FSF free software directory (2002): 87.9% GPL, 6.6% LGPL, 2.0% BSD or BSD-like, 1.9% Artistic, 0.3% MIT
  - MITRE DoD survey (2003): 52% GPL, 6% BSD, 5% Apache, 4% various "Community“, 3% LGPL
- Many big OSS projects changed to GPL-compatible
  - Python, vim, Mozilla, Zope, BSD, Apache (?), Qt, Wine, Alfresco
- XFree86 project died trying to become GPL-incompatible
- Avoid GPL-incompatible OSS licenses (risk of failure)
- Use common compatible licenses for new OSS projects
  - In particular: GPL, LGPL, MIT/X, or BSD-new

Unnecessary Fears

- Will OSS/FS destroy intellectual property? No.
  - Usually, complaint is about GPL
  - GPL trades you the right to freely incorporate their code into your software in exchange for the right to freely incorporate your code [which incorporates their code] into theirs
  - Intellectual property traded for other intellectual property
  - Microsoft sells GPL'ed software

Unnecessary Fears

- Viewing and changing source code valuable for non-programmers? Surprisingly, yes.
  - “Would you buy a car with the hood welded shut? If not, what do you know about modern … engine technology?” [Bob Young]
  - Consumers demand this so they can have control over their products, instead of dealers
- Anti-Microsoft campaign? No, not by all.
  - Jun02, 831 projects use Visual Basic; 8867 projects work on Windows [SourceForge]
  - Microsoft has been repeatedly asked to join community
  - Microsoft sells GPL'ed software

Acronyms

- COTS: Commercial Off-the-Shelf (either proprietary or OSS)
- DoD: Department of Defense
- HP: Hewlitt-Packard Corporation
- JTA: Joint Technical Architecture (list of standards for the DoD); being renamed to DISR
- OSDL: Open Source Development Labs
- OSS: Open Source Software
- RFP: Request for Proposal
- RH: Red Hat, Inc.
- U.S.: United States

Trademarks belong to the trademark holder.

Interesting Documents/Sites

- “Why OSS/FS? Look at the Numbers!” http://www.dwheeler.com/oss_fs_why.html
- “Use of Free and Open Source Software in the US Dept. of Defense” (MITRE, sponsored by DISA)
- President's Information Technology Advisory Committee (PITAC) -- Panel on Open Source Software for High End Computing, October 2000
- “Open Source Software (OSS) in the DoD,” DoD memo signed by John P. Stenbit (DoD CIO), May 28, 2003
- Center of Open Source and Government (EgovOS) http://www.egovos.org/
- OpenSector.org http://opensector.org
- Open Source and Industry Alliance http://www.osaia.org
- Open Source Initiative http://www.opensource.org
- Free Software Foundation http://www.fsf.org
- OSS/FS References http://www.dwheeler.com/oss_fs_refs.html