Open Source Software

The process, the outcome
The heat, the light

Ed Lazowska
IT & Public Policy
Autumn 2004

SOME RIGHTS RESERVED

The case for open source software

- "Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!"
  - David A. Wheeler
  - Revised as of September 30 2004
  - http://www.dwheeler.com/oss_fs_why.html

(Not the focus of this lecture, but an excellent source of information)

The open source software process

- "Two Case Studies of Open Source Software Development: Apache and Mozilla"
  - Audris Mockus, Roy T. Fielding, James D. Herbsleb
  - *ACM Transactions on Software Engineering and Methodology 11,3*
  - http://www-2.cs.cmu.edu/~jdh/collaboratory/research_papers/TOSEM-draft.pdf

Apache

- Process
  - Roles and responsibilties
    - "Apache Group" of core developers, elected by current members, can commit code to CVS; group votes on inclusion of major changes
  - Identifying work to be done
    - Issues are raised on bboards or BUGDB and discussed by the AG
  - Assigning and performing work
    - Once a problem or enhancement finds favor with the AG, a volunteer is found to work on it

- Prerelease testing
  - The developer tests – a "unit test" or "feature test," not a "regression test"
- Inspections
  - Core developers review all changes but not mandated
- Managing releases
  - Some core developer becomes the "release manager," driving issues to conclusion prior to a release

- Size of community
  - 400 individuals contributed code
    - 182 contributed to 695 fixes
    - 249 contributed 6,092 enhancements
  - 3,060 people submitted 3,975 problem reports
    - Only 591 reports led to code changes

Distribution of work: new functionality

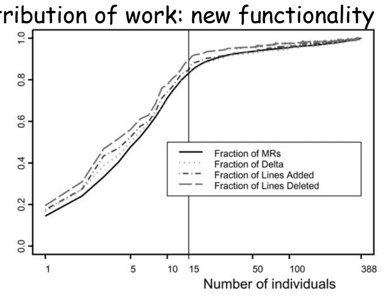


Fig. 1. The cumulative distribution of contributions to the code base.

Distribution of work: fixes

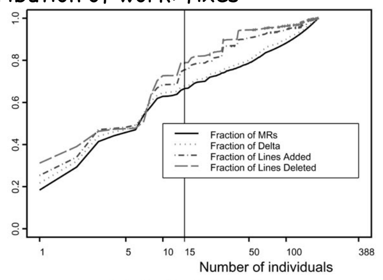


Fig. 2. Cumulative distribution of fixes.

Who reports problems?

- 3,975 distinct problem reports from 3,060 people
  - Top 15 reporters submitted only 5%
  - 2,600 submitted 1
  - 306 submitted 2
  - 85 submitted 3
  - Max was 12 submittals

Code ownership

- Broad-based contributions
- Of 42 .c files with more than 30 changes:
  - 40 had at least 2 developers making more than 10% of the changes
  - 20 had at least 4 developers making more than 10% of the changes

Defect density

Table III. Comparison of Defect Density Measures

Measure	Apache	A	C	D	E
Postrelease Defects/KLOCA	2.64	0.11	0.1	0.7	0.1
Postrelease Defects/KDelta	40.8	4.3	14	28	10
Postfeature test Defects/KLOCA	2.64	*	5.7	6.0	6.9
Postfeature test Defects/KDelta	40.8	*	164	196	256

Note: “Postfeature test” is “Postrelease” for Apache

Time to resolve issues

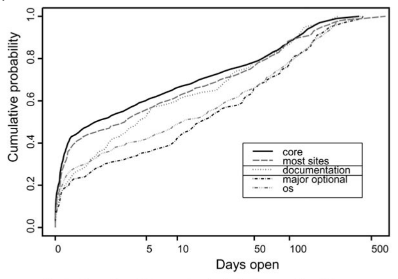


Fig. 4. Proportion of changes closed within given number of days.

Mozilla

- Process – many contrasts to Apache (Mozilla is hybrid)
  - Roles and responsibilties
    - Netscape "open-sourced" Communicator; project guided by the mozilla.org staff of 12, few of whom are active coders; there are some full-time professional developers, e.g., from Sun
  - Identifying work to be done
    - There is a "roadmap" document (plus bug reporting)
  - Assigning and performing work
    - Very loose approach to assigning/performing work

13

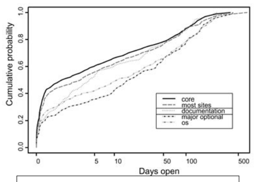
- Prerelease testing
  - mozilla.org performs a daily build
- Inspections
  - Two levels of formal code inspection for each change
- Managing releases
  - Continuous build process leads towards releases

14

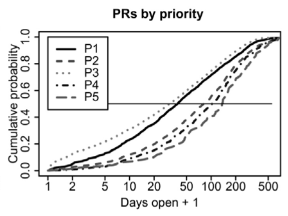
- Size of community
  - Much larger code base, so much larger change rate and much larger participating community, but "ratios" are roughly similar (e.g., % of reports that stimulated action)
- Distribution of work
  - Somewhat more uniform – more like commercial offerings than Apache
- Code Ownership
  - Code ownership is enforced – someone "owns" each module

15

- Time to resolve issues



Apache



Mozilla

16

Hypotheses

1. There will be a core of developers who control the code base and create >=80% of new functionality. If coordination is ad hoc, this group will be <=15 people.
2. If size of project requires a core >15, there will need to be explicit development processes, code ownership, required inspections
3. A group larger by an order of magnitude will repair defects, and a group larger by another order of magnitude will report bugs

17

4. Projects with a strong core but a small community will add features but fail because of failure to find and fix bugs.
5. Defect density post-feature-test will be lower than commercial code
6. Developers will be users
7. Rapid response to customer problems
   - *Not the case* for Mozilla!

18

Contrast to the proprietary code development process

- Frequency of releases
- Frequency of patches
- Standards for release
- Testing procedures
- Ease of prototyping

19

Final comment on OSS Process

- "A Second Look at the Cathedral and the Bazaar"
  - Nikolai Bezroukov
  - First Monday, 1999
  - http://www.firstmonday.dk/issues/issue4_12/bezroukov/
  - [more on *The Cathedral and the Bazaar* later]

20

- "There are advantages in using mixed models other than the pure centralized (Cathedral) or completely decentralized (Bazaar) extremes. It's hardly surprising that in reality a mixed model dominates or that there's a place for highly centralized development in the Linux world."
- Linus Torvalds: "Open source may sound democratic, but it isn't. At the LinuxWorld Expo on Wednesday, leaders of some of the best-known open source development efforts said they function as dictators."

21

Open source software and reliability/security: Opinion

- The $64,000 (,000? ,000,000?) question: How does the reliability/security of open source code compare to that of proprietary code?
  - Argument for superiority:
    - Linus Torvalds [attributed, in *The Cathedral and the Bazaar*]: "Given enough eyeballs, all bugs are shallow."
    - *The Cathedral and the Bazaar*
      - Eric Raymond
      - O'Reilly, 1/15/02 (revised edition)
      - http://www.catb.org/~esr/writings/cathedral-bazaar/

22

- Counter-argument:
  - Gene Spafford: "The open-source movement is largely devoid of systematic efforts to guarantee security. The fact that code *can* be examined for flaws does not mean it *will* be examined by anyone competent."
- Funnier counter-argument:
  - Albert Einstein [quoted in "A Second Look at the Cathedral and the Bazaar"]: "Only two things are infinite, the universe and human stupidity, and I'm not sure about the former."

23

- Middle ground:
  - Linus Torvalds: "People think just because it is open-source, the result is going to be automatically better. Not true. You have to lead it in the right directions to succeed. Open source is not the answer to world hunger."
  - Gene Spafford: "Careful analysis leads to the conclusion that security is unrelated to whether the software is proprietary or open source … The truth is that neither the open-source nor the proprietary paradigms offer any kind of silver bullet for security and quality."

24

- "Whether Linux or Windows, No Software Is Secure"
  - Eugene H. Spafford and David L. Wilson
  - Chronicle of Higher Education, 9/24/04
  - http://www.cs.washington.edu/education/courses/csep590tu/04au/readings/insecure.htm
- A few (approximate) quotes
  - Claims and counterclaims … miss the main point: Today's computer systems, whether open source or proprietary, are inherently insecure because of inconsistent and haphazard design, lack of interest in ensuring high quality, and a marked indifference on the part of developers to the growing complexity of systems.

- Careful analysis leads to the conclusion that security is unrelated to whether the software is proprietary or open source.
- The open-source movement is largely devoid of systematic efforts to guarantee security. The fact that code *can* be examined for flaws does not mean it *will* be examined by anyone competent.
- The literature contains reports of serious security flaws in open-source products, often after years of use. Several occurred in the parts of the software intended to make it secure, which presumably underwent more careful coding and examination. That strongly suggests that either the many people who supposedly look at the code are not able to recognize the problems, or they aren't really looking. Experience indicates that both are true.

- The truth is that neither the open-source nor the proprietary paradigms offer any kind of silver bullet for security and quality.
- Until we focus on applying sound security technology, on appropriately training the people who produce the programs, and on paying more attention to the quality of software than to its number of features and purchase price, we will continue to experience problems with security.

- "Open Source Security: Still a Myth"
  - John Viega (co-author of *Secure Programming Cookbook for C and C++*)
  - O'Reilly, 9/16/04
  - http://www.onlamp.com/pub/a/security/2004/09/16/open_source_security_myths.html
- A few (approximate) quotes:
  - Most people look for the low-hanging fruit: straightforward instances of common problems such as buffer overflows, format string problems, and SQL injection. Less sexy risks tend to get ignored.

- Just looking for the common problems can be incredibly difficult and time consuming. For instance, even though buffer overflows are a well-understood, straightforward problem, in plenty of instances they've remained in heavily audited code for years.
- The commercial world has better analysis tools available. (Clearly, "eyeballs aren't enough"!)
- Customer pressure is starting to have a big impact on development processes. For example, for the past two years Microsoft has made a dramatic effort toward improving software security throughout the organization.
- Open source can prevail, but needs:
  - Process
  - Security awareness across the board
  - Independent, third-party auditing

- Ken Thompson, 1999, in *Computer:*
  - "I view Linux as something that's not Microsoft - a backlash against Microsoft, no more and no less … I've looked at the source and there are pieces that are good and pieces that are not. A whole bunch of random people have contributed to this source, and the quality varies drastically."

- "Is it harmful to discuss security vulnerabilities?"
  - Matt Blaze
  - http://www.crypto.com/hobbs.html
- General thrust: "security through obscurity" is nonsense
  - You can't rely on keeping the code secret
  - It makes no sense to suppress discussion of security flaws
- "The debate over the open discussion of security vulnerabilities long predates the Internet and computers."

31

- A.C. Hobbs, *Locks and Safes: The Construction of Locks*, 1853: "A commercial, and in some respects a social doubt has been started within the last year or two, whether or not it is right to discuss so openly the security or insecurity of locks. Many well-meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty, by showing others how to be dishonest. This is a fallacy. Rogues are very keen in their profession, and know already much more than we can teach them respecting their several kinds of roguery."

32

Open source software and reliability/security: Fuzz testing

- "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services"
  - Barton P. Miller, et al., 1995
  - ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-revisited.pdf

33

- Methodology
  - Subjected 80 UNIX utilities to random input streams
  - Ditto for network services
  - Ditto for X-window applications
  - Utilities from 7 commercial UNIX systems, and 2 OSS systems

34

- Findings
  - Failure rate on tests is 18-23%
    - "Failure" = crashing or hanging
  - Failure rate of utilities on commercial versions of UNIX (from Sun, IBM, SGI, DEC, and NeXT) is 15-43%
  - Linux: 9%
  - GNU: 6%
  - >50% of X-window aps crashed on random input; >25% crashed on random but *legal* X-event streams

35

- Sources of crashes/hangs
  - Pointers/arrays
  - Dangerous input functions
  - Signed characters
  - End-of-file

36

- "An Empirical Study of the Robustness of Windows NT Applications Using Random Testing"
  - Forrester and Miller, 2000
  - ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz-nt.pdf

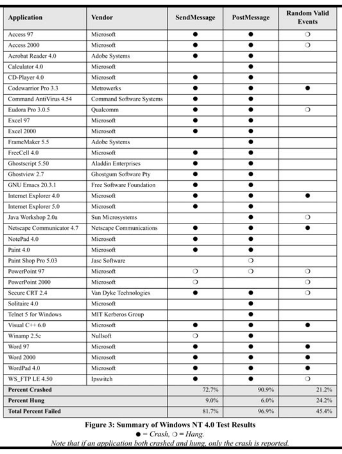
37

- Methodology
  - Subjected 33 NT 4 (and 14 NT 5) application programs utilities to random input using `SendMessage`, `PostMessage`, `keybd_event`, and `mouse_event`

38

- Findings
  - 21% of aps crashed and 24% hung when presented with random *valid* keyboard and mouse events
  - 100% failed when presented with completely random Win32 messages

39



Application	Vendor	SendMessage	PostMessage	Random Valid Events
Access 97	Microsoft	●	●	❍
Access 2000	Microsoft	●	●	❍
Acrobat Reader 4.0	Adobe Systems	●	●	
Calculator 4.0	Microsoft		●	
CD-Player 4.0	Microsoft	●	●	
Codewarrior Pro 3.3	Metrowerks	●	●	●
Command AntiVirus 4.54	Command Software Systems	●	●	
Eudora Pro 3.0.5	Qualcomm	●	●	❍
Excel 97	Microsoft	●	●	
Excel 2000	Microsoft	●	●	
FrameMaker 5.5	Adobe Systems		●	
FreeCell 4.0	Microsoft	●	●	
Ghostscript 5.50	Aladdin Enterprises	●	●	
Ghostview 2.7	Ghostgum Software Pty	●	●	
GNU Emacs 20.3.1	Free Software Foundation	●	●	
Internet Explorer 4.0	Microsoft	●	●	●
Internet Explorer 5.0	Microsoft	●	●	
Java Workshop 2.0a	Sun Microsystems		●	❍
Netscape Communicator 4.7	Netscape Communications	●	●	●
NotePad 4.0	Microsoft	●	●	
Paint 4.0	Microsoft	●	●	
Paint Shop Pro 5.03	Jasc Software		❍	
PowerPoint 97	Microsoft	❍	❍	❍
PowerPoint 2000	Microsoft	❍		❍
Secure CRT 2.4	Van Dyke Technologies	●	●	❍
Solitaire 4.0	Microsoft		●	
Telnet 5 for Windows	MIT Kerberos Group		●	
Visual C++ 6.0	Microsoft	●	●	●
Winamp 2.5c	Nullsoft	❍	●	
Word 97	Microsoft	●	●	●
Word 2000	Microsoft	●	●	●
WordPad 4.0	Microsoft	●	●	●
WS_FTP LE 4.50	Ipswitch	●	●	❍
Percent Crashed		72.7%	90.9%	21.2%
Percent Hung		9.0%	6.0%	24.2%
Total Percent Failed		81.7%	96.9%	45.4%

Figure 3: Summary of Windows NT 4.0 Test Results
● = *Crash,* ❍ = *Hang.*
Note that if an application both crashed and hung, only the crash is reported.

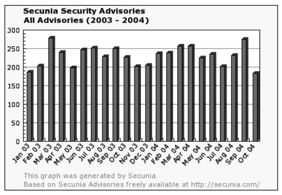
- Sources of crashes/hangs
  - Only had source code for 2 applications, so analysis limited
  - Common careless programming idiom: Receiving a Win32 message and unsafely using a pointer or handle contained in the message

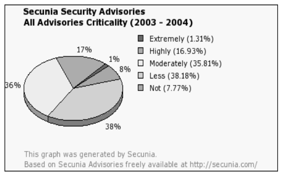
43

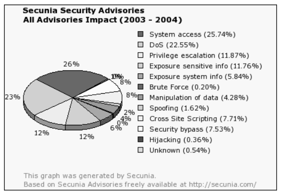
Open source software and reliability/security: Secunia

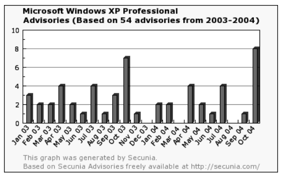
- An *incredible* wealth of data!
  - http://secunia.com/

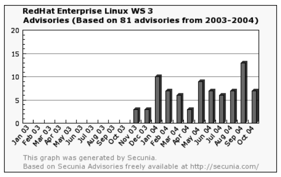
44

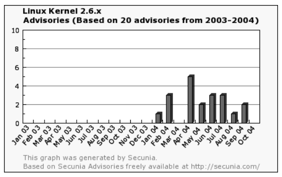


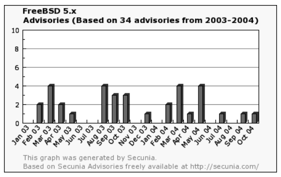


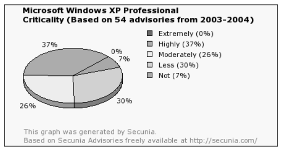


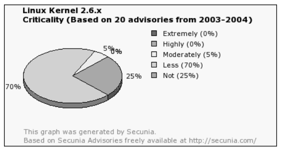


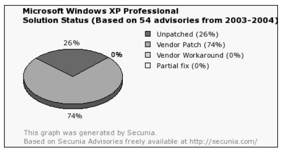


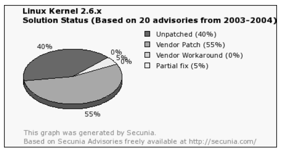












Open source software and reliability/security: Browsers

- Recent ISS SecurityFocus posting, covered on Slashdot 10/19/04
  - "Web browsers – a mini-farce"
  - Michal Zalewski
  - http://www.securityfocus.com/archive/1/378632/2004-10-15/2004-10-21/0

48

- Methodology
  - Tool generates “tiny, razor-sharp shards of malformed HTML” – only basic HTML
  - Ran against IE, Mozilla/Nescape/Firefox, Opera, Lynx, Links
- Results
  - “All browsers but Microsoft Internet Explorer kept crashing on a regular basis due to NULL pointer references, memory corruption, buffer overflows, sometimes memory exhaustion”

- Examples
  - Mozilla: Memory corruption / overflow causes null pointer access
  - Mozilla: Bogus pointer access
  - Opera: Referenced un-initialized memory
  - Links: Big table consumes all memory then over-writes what it managed to allocate
  - Lynx: Loops forever trying to render broken HTML

- Conclusion
  - “Only MSIE appears to be able to consistently handle malformed input well, suggesting this is the only program that underwent rudimentary security QA testing with a similar fuzz utility.”
  - “This is of course not to say MSIE is more secure; it does have a number of problems, mostly related to its security architecture and various features absent in other browsers. But the quality of core code appears to be far better than of its "secure" competitors.”

OSS Summary

- A small core group of developers control the code base and create most of the new functionality
- A group larger by an order of magnitude repairs defects, and a group larger by another order of magnitude reports bugs
- Many hybrid models exist
- “Open source may sound democratic, but it isn’t.” [Linus Torvalds]

- “Claims and counterclaims … miss the main point: Today's computer systems, whether open source or proprietary, are inherently insecure.” [Gene Spafford]
- “Careful analysis leads to the conclusion that security is unrelated to whether the software is proprietary or open source … The truth is that neither the open-source nor the proprietary paradigms offer any kind of silver bullet for security and quality.” [Gene Spafford]