Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore

Ross Anderson

Cambridge University, England; email `rja14@cl.cam.ac.uk`

Abstract. Some members of the open-source and free software community argue that their code is more secure, because vulnerabilities are easier for users to find and fix. Meanwhile the proprietary vendor community maintains that access to source code rather makes things easier for the attackers. In this paper, I argue that this is the wrong way to approach the interaction between security and the openness of design. I show first that under quite reasonable assumptions the security assurance problem scales in such a way that making it either easier, or harder, to find attacks, will help attackers and defendants equally. This model may help us focus on and understand those cases where some asymmetry is introduced.

However, there are more pressing security problems for the open source community. The interaction between security and openness is entangled with attempts to use security mechanisms for commercial advantage – to entrench monopolies, to control copyright, and above all to control interoperability. As an example, I will discuss TCPA, a recent initiative by Intel and others to build DRM technology into the PC platform. Although advertised as providing increased information security for users, it appears to have more to do with providing commercial advantage for vendors, and may pose an existential threat to open systems.

1 Introduction

It is frequently argued in the open source and free software community that making source code available to all is good for security. A large community of users and experts can pore over the code and find vulnerabilities: 'to many eyes, all bugs are shallow' [1]. In the crypto community in particular, it has been standard practice since the nineteenth century that the opponent knows the design of your system, so the only way you can keep him out is by denying him knowledge of a temporary variable, the key [2]. On the other hand, opponents of open source software argue that 'if the software is in the public domain, then potential hackers have also had the opportunity to study the software closely to determine its vulnerabilities' [3]. So whom does openness help more, attack or defence?

This appears to be the kind of question that is in principle amenable to a clear answer, on the basis of accepted models of software reliability growth, empirical data about reported vulnerabilities, or (preferably) both.

The question is much more general than whether software source code should be available to users. A wide range of systems and components can be either easier, or more difficult, to test, inspect and repair depending on the available tools and access. Object code is partly accessible for inspection, through debugging and disassembly tools, but may be difficult to patch. Hardware devices, even those designed to be tamper-proof, can often be reverse engineered with surprisingly little effort – although the capital resources needed to fabricate a compatible clone might be scarce. I also do not wish to take sides in the 'open source' versus 'free software' debate. So in what follows I will consider 'open systems' versus 'closed systems', which differ in the degree of difficulty in finding a security vulnerability.

2 Security Reliability Growth

Safety-critical software engineers have known for years that for a large, complex system to have a mean time before failure (MTBF) of (say) 100,000 hours, it must be subject to at least that many hours of testing [4]. This was first observed by Adams in 1985 in a study of the bug history of IBM mainframe operating systems [5], and has been confirmed by extensive empirical investigations since. The first theoretical model explaining it was published in 1996 by Bishop and Bloomfield [6]. Such reliability growth models were developed for software reliability in general, but they can be applied to bugs of any particular subtype – such as defects that might cause loss of life, or loss of mission, or the breach of a security policy. We only require that there are enough bugs for statistical arguments to work, and that a consistent definition of 'bug' is used throughout.

When we test software, we first find the most obvious bugs – that is, the bugs with the lowest mean time to failure. After about ten minutes, we might find a bug with the 10-minute MTBF. Then after half an hour we might get lucky and find a bug with an MTBF of 42 minutes, and so on. In a large system, of course, luck cancels out and we can use statistics. A hand-waving argument would go that, after a million hours of testing, we'd have found all the bugs with an MTBF of less than a million hours, and we'd hope that the software's overall reliability would be proportional to the effort invested.

Reliability growth models seek to make this more precise. Suppose that the probability that the i-th bug remains undetected after t random tests is $e^{-E_i t}$. The models cited above show that, after a long period of testing and bug removal, the net effect of the remaining bugs will under certain assumptions converge to a polynomial rather than exponential distribution. In particular, the probability E of a security failure at time t, at which time n bugs have been removed, is

$$E = \sum_{i=n+1}^{\infty} e^{-E_i t} \approx K/t$$

over a wide range of values of t. I give in the appendix a brief proof of why this is the case, following the analysis by Brady, Anderson and Ball [7]. For present purposes, note that this explains the slow reliability growth observed in practice. The failure time observed by a tester depends only on the initial quality of the code (the constant of integration K) and the time spent testing it so far.

Consider now what happens if we make the tester's job harder. Suppose that after the initial alpha testing of the product, all subsequent testing is done by beta testers who have no access to the source code, but can only try out various combinations of inputs in an attempt to cause a failure. If this makes the tester's job on average λ times harder, so the bugs are λ times more difficult to find, and the probability that the i-th bug remains undetected becomes $e^{-E_i \lambda t}$, then the probability that the system will fail the next test is

$$E = \sum_{i=n}^{\infty} e^{-E_i t/\lambda} \approx K/\lambda t$$

In other words, the system's failure rate has just dropped by a factor of λ, just as we would expect.

However, what if all the testing to date had been carried out under the more difficult regime? In that case, only $1/\lambda$ the amount of effective testing would have been carried out, and the λ factors would cancel out. Thus the failure probability E would be unchanged.

Going back to our intuitive argument, making bugs seven times more difficult to find will mean that we now work over an hour to find the bug whose MTBF was previously 10 minutes, and over four hours for the 42-minute bug. But the reliability of software still grows as the time spent testing, so if we needed 10,000 hours of testing to get a 10,000-hour-MTBF product before, that should still hold now. We will have removed a smaller set of bugs, but the rate at which we discover them will be the same as before.

Similar considerations apply where proprietary software is first tested by insiders with access to source code. With a large commercial product, dozens of testers may work for a year on this code, after which it will go out for beta testing by outsiders whose access is to the object code only. There might be tens of thousands of beta testers, so even if λ were as large as 100, the effect of the initial, open, alpha-testing phase will be quickly swamped by the very much greater overall effort of the beta testers.

A vendor of proprietary software may have exogenous reasons for not making source code available (a common argument is that he might fear an avalanche of vexatious lawsuits by people holding software patents with no intrinsic merit but who hope to extract a settlement by threatening expensive litigation). Then a straighforward economic analysis can in principle tell us the right time to roll out a product for beta testing. Alpha testers are more expensive, being paid a salary; as time goes on, they discover fewer bugs and so the cost per bug discovered climbs steadily. At some threshold, perhaps once bug removal starts to cost more than the damage that bugs could do in a beta release product, alpha testing stops. Beta testing is much cheaper; testers are not paid (but may get discounted software, and still incur support costs). Eventually – in fact, fairly quickly – the beta test effort comes to dominate reliability growth.

So, other things being equal, we expect that open and closed systems will exhibit similar growth in reliability and in security assurance.

This does not of course mean that, in a given specific situation, proprietary and open source are evenly matched. But we have to look at second-order effects, asymmetries, transients and nonlinear effects to determine which is better where. This is where we expect the interesting economic and social effects to be found.

3 Symmetry Breaking

Even though open and closed systems are equally secure in an ideal world, the world is not ideal, and is often adversarial. We can expect attackers to search for, find and exploit phenomena that break the symmetry between open and closed models.

3.1 Transients and transaction costs

Transient effects may matter. Suppose that a new type of abstract attack is found by an academic researcher and published. It may be simple to browse the GNU/Linux source code to see if it can be applied, but much more complex to construct test cases, write debugging macros etc, to see if an exploit can be made for Windows. So there may be time-to-market issues for the attack.

There may also be time-to-market issues for the defence. Transaction costs do matter; Adams reported that IBM fixed mainframe operating system bugs the eighth time they were reported [5], while Leung has studied the optimal frequency of security updates from the customer perspective [8].

Transaction costs may persuade some vendors to remain closed. For example, if source code were made available to beta testers too, then the initial reliability of beta releases would be worse, as the testers would be more efficient. Fairly soon, the reliability would stabilise at the staus quo ante, but a much larger number of bugs would have had to be fixed by the vendor's staff. Avoiding this cost might well be a rational argument against open source.

3.2 Behaviour of the vendor

There is already some discussion about whether opening the source of a product makes the programmers more likely to write better code – code that is less complex, smaller and more maintainable. This could certainly influence the value of k in the above equation. However there are also incentive issues.

The motivation of the vendor to implement fixes for reported bugs can be affected in practice by many factors. The US government prefers vulnerabilities in some products to be reported to authority first, so that they can be exploited by law enforcement or intelligence agencies for a while. Vendors are only encouraged to ship patches once outsiders start exploiting the hole too.

In addition, our team has long experience of security hardware and software vendors preferring to keep quiet about bugs, and only shipping patches once their hand is forced (e.g., by TV publicity). They may feel that shipping a patch undermines previous claims of absolute protection, and even if this is not company policy, managers may not want to undermine assurances previously given to their bosses. So there may be information asymmetries and principal-agent effects galore. The argument is now swinging in favour of policies of vulnerability disclosure after a fixed notice period; without the threat of eventual disclosure, little may get done. (This is not going to be a panacea, though; on at least one occasion, a nine-month grace period that we gave a vendor before publication was consumed entirely by internal wrangling about which department was to blame for the flaw.)

3.3 Test focus, and the Wild West

Another set of issues have to do with test focus. The models discussed above assume that testing is random, yet in practice a tester is likely to focus on a particular subset of test cases that are of interest to her or are easy to perform with her equipment.

Experienced testers know that most bugs are to be found in recently added code, and will focus on this. In fact, one real advantage that source code access gives to an attacker is that it makes it easier to identify new code. This does not affect our argument, though, as the effects are subsumed into the value of λ.

However, the individual preferences and skills of testers still vary. It is well known that software may be tested extensively by one person, until it appears to be very reliable, only to show a number of bugs quickly when passed to a second tester. This is already recognised as an economic argument for parallelism in testing [7]. It is also an argument for extensive beta testing; a large set of testers is more likely to be representative of the ultimate user community.

A related argument for closed systems is as follows. Think of the Wild West; the bandits can concentrate their forces to attack any bank on the frontier, while the sherriff's men have to defend everywhere. Now, the level of assurance of a given component is a function of the amount of scrutiny that it actually gets, not of what it might get in theory. As testing is boring, and volunteers generally only want to fix failures that irritate them, the amount of concentrated attention paid by random community members to (say) the smartcard device drivers for GNU/Linux is unlikely to match what an enemy government might invest [10]. A counter-argument can be drawn from Benkler's model, that large communities can include individuals with arbitrarily low reservation prices for all sorts of work [11]. A different one arises in the context of our reliability growth model; efficacy of focus appears to assume that the attacker is more efficient than the defender at selecting a subset of the code to study for vulnerabilities; if they were randomly distributed, then no one area of focus should be more productive for the attacker than any other.

The more relevant consideration for security assurance is, I believe, the one in [12] – that a large number of low-probability bugs structurally favours attack over defence. In an extreme case, a system with 10^6 bugs each with an MTBF of 10^9 hours will have an MTBF of 1000 hours, so it will take about that much time to find an attack. But a defender who spends even a million hours has very little chance of finding that particular bug before the enemy exploits it. This problem was known in generic terms in the 1970s; the above model makes it more precise.

3.4 Do defenders cooperate or free-ride?

How eager will a product's users be to report bugs? One might expect that, on ideological grounds, people would be more willing to report bugs to the maintainers of a cooperative software project such as Linux or Apache than to the owners of a proprietary product such as Windows. (I am ignoring such transaction cost issues such as the difficulty of getting through to a helpline.)

But it is not quite that simple. In the paper that follows this one at the workshop, Varian presents an interesting analysis of how defenders are likely to react when the effectiveness of their defence depends on the sum total of all their efforts, the efforts of the most energetic and efficient defender, or the efforts of the least energetic and efficient defender [13]. In the total-efforts case, there is always too little effort exerted at the Nash equilibrium as opposed to the optimum, but at least reliability continues to increase with the total number of participants. This kind of analysis looks like a fruitful way forward.

I will conclude this section by stating that although there is no difference in security assurance growth in the ideal case between open and closed systems, there are enough deviations from the ideal for the choice to be an important one and a suitable subject for researchers in the economics of information security. It would be particularly interesting if there were any studies of reliability growth for code that has bifurcated, and has an ecnumbered and non-encumbered version.

However, assurance growth is not the only, or even the main, issue with open systems and security.

4 Real World Problems

Information security is about money and power; it's about who gets to read, write, or run which file. The economics of information goods and services markets is dominated by the establishment and defence of monopolies, the manipulation of switching costs, and the control of interoperability [14]. It should surprise no-one that the most rapidly growing application area of information security mechanisms is in the support of such business plays. For example, some mobile phone vendors use challenge-response authentication to check that the phone battery is a genuine part rather than a clone – in which case, the phone will refuse to recharge it, and may even drain it as quickly as possible. The Sony Playstation 2 uses similar authentication to ensure that memory cartridges were made by Sony rather than by a low-price competitor – and the authentication chips also contain the CSS encryption algorithm for DVD, so that reverse engineers can be accused of circumventing a copyright protection mechanism and hounded under the Digital Millennium Copyright Act.

4.1 TCPA

In this brave new world, it is surprising that so few have paid attention to the Trusted Computing Platform Alliance (TCPA), an initiative led by Intel whose stated goal is to embed digital rights management technology in the PC [15].

TCPA provides for a monitoring component to be mounted in future PCs. The standards document is agnostic about whether this is in the CPU (the original Intel idea), the O/S (the Microsoft idea) or a smartcard chip or dongle soldered to the motherboard. For simplicity, I'll assume the last of these, and call the chip 'Fritz' for brevity, in honour of Senator Hollings, who is working tirelessly in Congress to make TCPA a mandatory part of all consumer electronics.

When you boot up your PC, Fritz takes charge. He checks that the boot ROM is as expected, executes it, measures the state of the machine; then checks the first part of the operating system, loads and executes it, checks the state of the machine; and so on. The trust boundary, of hardware and software considered to be known and verified, is steadily expanded. A table is maintained of the hardware (audio card, video card etc) and the software (O/S, drivers, etc); if there are significant changes, the machine must be re-certified.

The result is a PC booted into a known state with an approved combination of hardware and software [16]. Once the machine is in this state, Fritz can prove it to third parties: for example, he will do an authentication protocol with Disney to prove that his machine is a suitable recipient of 'Snow White'. The Disney server then sends encrypted data, with a key that Fritz will use to unseal it. Fritz makes the key available only so long as the environment remains 'trustworthy'.

The rules governing which application can use which data can be quite expressive – and imposed remotely. Music and video vendors can sell content only to those PCs they trust not to rip it. In government applications, I expect the idea is to implement a version of Bell-LaPadula, so that government information can 'leak' only through a small number of trusted subjects; if you are not so trusted, you will be unable to post a file containing classified information to a journalist, as his Fritz will not give him the necessary key.

4.2 The digital commons and the threat to open systems

There are potentially serious issues for consumer choice and for the digital commons [17]. If TCPA-compliant PCs become the majority, and if TCPA-enabled applications prevail in the marketplace, then I expect there will be another segmentation of the platform market similar to that between Linux and Windows now. If you boot up your PC in an untrusted state, it will be much like booting it with Linux now: the sound and graphics will not be so good, and there will not be nearly so many applications. Worse, while at present it can be slightly tiresome for a Linux user to read Microsoft format documents, under TCPA Microsoft can prevent this completely: rather than messing around with file formats so that existing competitor products have to be upgraded before they can read them, Microsoft (or any other application owner) can cause data to be encrypted using TCPA keys whose export they control completely.

When pressed on TCPA, the explanation from Intel managers is that they want the PC to be the central information appliance in the home, acting as a hub for the CD player, the TV, the fridge and everything else. If DRM becomes an intergal part of entertainment, and entertainment's the main domestic app, then the PC must do DRM, or the set-top-box might take over from it. (Other vendors have different stories.)

TCPA raises many issues. What are the governance arrangements of the Alliance? How will its products and policies interact with the new copyright regulations due in EU countries under the recent Copyright Directive? Britain, for example, seems reluctant to impose any duties on the vendors and beneficiaries of DRM technology, while eager to smooth their path by removing as many of the existing fair use exemption as the EU will permit. The sort of practical question we are likely to be faced with this year is whether the blind will still be able to use their screen scrapers to read e-books. And what about the transparency of processing of personal data enshrined in the EU data protection directive? What about the simple sovereignty issue, of who will write copyright regulations in Europe in future – will it be the European Commission assisted by national governments, as at present, or an application developer in Portland or Redmond? And will TCPA be used by Microsoft as a means of killing off competitors such as GNU/Linux and Apache, which being maintained by networks of volunteers cannot simply be bought out and closed down?

4.3 Competition policy issues

Perhaps the most serious issues on the macro scale, though, have to do with competition policy and the development of markets for information goods and services. If the owner of an application has total control in future over who can use the data users generate with it, and change the rules remotely, then this can have many serious effects. Compatibility between applications will be able to be controlled remotely, and with a subtlety and strength of mechanism that has never been available before. So it creates the power to magic all sorts of new monopolies into existence, and to abolish the right to reverse engineer for compatibility. The implications of this are broad; for discussion, see for example Samuelson and Scotchmer [18].

The conventional antitrust approach has been to smile on standards developed in open fora, while being suspicious of proprietary standards. This has already shown itself to be inadequate in the 1930s, when IBM and Remington Rand maintained a choke-hold on the punched card market by patent licensing agreements. Since then a significant literature has developed [19, 20].

The Intel modus operandi appears to take this to new heights. Gawer and Cusumano describe how Intel honed a 'platform leadership' strategy, in which they led a number of industry efforts – the PCI bus, USB and so on [21]. The positive view of this strategy was that they grew grow the overall market for PCs; the dark side was that they prevented any competitor achieving a dominant position in a technology that might have threatened Intel's dominance of the PC hardware. Thus, Intel could not afford for IBM's microchannel bus to prevail, not just as a competing nexus of the PC platform but also because IBM had no interest in providing the bandwidth needed for the PS to compete with high end systems. Presumably also Intel did not want one of the security vendors to establish leadership in DRM technology once it seemed plausible that DRM could become the key component for home computing: the home platform should be a "PC" rather than a "device compliant with Acme's DRM standard".

Intel's modus operandi is to set up a consortium to share the development of the technology, have the founder members of the consortium put some IP into the pot, publish a standard, get some momentum behind it, then license it to the industry on the condition that licensees in turn cross-license any interfering IP of their own, at zero cost, to all corsortium members. The effect in strategic terms is somewhat similar to the old Roman practice of demolishing all dwellings and cutting down all trees within two bowshots of a road, or half a mile of a castle. No competing structure may be allowed near Intel's platform; it must all be levelled into a commons. But a nice, orderly, well-regulated commons: interfaces should be 'open but not free'.

Maybe one can see the modus operandi used with TCPA (and PCI bus, and USB) as the polished end result of evolution - of a package of business methods that enable a platform leader to skirt antitrust law. It also diffuses responsibility, so that when TCPA eventually does get rolling there is no single player who takes all the media heat – as Intel earlier did with the Pentium serial number.

TCPA is not vapourware. The first specification was published in 2000, IBM sells laptops that are claimed to be TCPA compliant, and some of the features in Windows XP and the X-Box are TCPA features. For example, if you change your PC configuration more than a little, you have to reregister all your software with Redmond.

What will be its large-scale economic effects?

Suppose you are developing a new speech recognition product. If you TCPA-enable it, then on suitable platforms you can cause its output to be TCPA-protected, and you can remotely decide what applications will be able to read these files, and under what conditions. Thus if your application becomes popular, you can control the complementary products and either spawn off a series of monopolies for add-ons, or rent out access to the interfaces, as you wish. The gains to be made by a company that successfully establishes a strong product are greatly enhanced. I expect that venture capitalists will insist that their investments be TCPA-protected. However, as the few winners win bigger, the proportion of winners overall may diminish, and as interfaces to existing data become rented rather than free, the scope for low-budget innovators to create new information goods and services rapidly will be severely circumscribed.

In other words, TCPA appears likely to change the ecology of information goods and services markets so as to favour incumbents, penalise challengers, and slow down the pace of innovation and entrepreneurship. It is also likely to squeeze open systems, and may give rise to serious trade disputes between the USA and the EU.

5 Conclusion

The debate about open versus closed systems started out in the nineteenth century when Auguste Kerckhoffs pointed out the wisdom of assuming that the enemy knew one's cipher system, so that security could only reside in the key. It has developed into a debate about whether access to the source code of a software product is of more help to the defence, because they can find and fix bugs more easily, or to attackers, because they can develop exploits with less effort.

This paper answers that question. In a perfect world, and for systems large and complex enough for statistical methods to apply, the attack and the defence are helped equally. Whether systems are open or closed makes no difference in the long run. The interesting questions lie in the circumstances in which this symmetry can be broken in practice.

This leads naturally to a second point. The interaction between security and the openness of systems is much more complex than just a matter of reliability. The heart of the matter is functionality – what should a secure system do? What does security mean to a platform vendor?

Seen in these terms, the answer is obvious. While security for the user might mean the repulse of 'evil hackers on the Internet', whoever they might be, security for the vendor means growing the market and crushing the competition. The real future tensions between the open and closed system communities will be defined by struggles for power and control over standards. TCPA adds a new twist to the struggle. Even if data standards achieve 'XML heaven' of complete openness and interoperability, the layer of control will shift elsewhere. Instead of being implicit in deviously engineered file format incompatibilities, it will be overtly protected by storng cryptography and backed up by the legal sanctions of DMCA.

Although TCPA is presented as a means of improving PC security and helping users protect themselves, it is anything but. The open systems community had better start thinking seriously about its implications, and policymakers should too.

Acknowledgements: Thanks for useful comments to Richard Clayton, Paul Leach, Hal Varian, Pater Wayner, Fabien Petitcolas, Brian Behlendorf, Seth Arnold, Jonathan Smith, Tim Harris and Andrei Serjantov. Mike Roe commented that this paper could be split into two, one on reliability growth and the second on TCPA. This may be fair comment; however, I'd accepted an invitation from the Open Source Software Economics conference to give an opening talk on TCPA, and the previous week I discovered the result described in the first part of this paper, which is just too appropriate not to include.

References

1. ES Raymond, "The Cathedral and the Bazaar", 1998, at `http://tuxedo.org/~esr/writings/cathedral-bazaar/`
2. A Kerckhoffs, "La Cryptographie Militaire", in *Journal des Sciences Militaires*, 9 Jan 1883, pp 5–38; `http://www.cl.cam.ac.uk/users/fapp2/kerckhoffs/`
3. K Brown, "Opening the Open Source Debate", 2002, Alexis de Toqueville Institution, at `http://www.adti.net/html_files/defense/opensource_whitepaper.pdf`
4. RW Butler, GB Finelli, "The infeasibility of experimental quantification of life-critical software reliability", *ACM Symposium on Software for Critical Systems*, New Orleans ISBN 0-89791-455-4 pp 66–76 (Dec 1991)
5. Adams E. N., *Optimising preventive maintenance of software products*, *lBM Journal of Research & Development*, Vol. 28, issue 1 pp 2–14 (1984)
6. P Bishop, R Bloomfield, "A Conservative Theory for Long-Term Reliability-Growth Prediction', *IEEE Transactions on Reliability* v 45 no 4 (Dec 96) pp 550–560
7. RM Brady, RJ Anderson, RC Ball, "Murphy's law, the fitness of evolving species, and the limits of software reliability", Cambridge University Computer Laboratory Technical Report no. 471 (September 1999), available at `http://www.cl.cam.ac.uk/ftp/users/rja14/babtr.pdf`
8. KS Leung, Diverging economic incentives caused by innovation for security updates on an information network, available at ¡http://www.sims.berkeley.edu/resources/affiliates/workshops/econsecurity/¿
9. Rain Forest Puppy, "Issue disclosure policy v1.1", at `http://www.wiretrip.net/rfp/policy.html`
10. M Schaefer, *Panel comments at Oakland 2001*
11. Y Benkler, "Coase's Penguin, or, Linux and the Nature of the Firm", at *Conference on the Public Domain*, Nov 9-11, Duke Law School, available at `http://www.law.duke.edu/pd/papers/Coase's_Penguin.pdf`
12. RJ Anderson, "Why Information Security is Hard – An Economic Perspective", in *Proceedings of the Seventeenth Computer Security Applications Conference*, IEEE Computer Society Press (2001), pp 358–365; available at `http://www.cl.cam.ac.uk/ftp/users/rja14/econ.pdf`

13. H Varian, “System Reliability and Free Riding”, available at `http://www.sims.berkeley.edu/resources/affiliates/workshops/econsecurity/econws/49.pdf`
14. C Shapiro, H Varian, *‘Information Rules’*, Harvard Business School Press (1998), ISBN 0-87584-863-X
15. `http://www.trustedpc.org`
16. WA Arbaugh, DJ Farber, JM Smith, “A Secure and Reliable Bootstrap Architecture”, in *Proceedings of the IEEE Symposium on Security and Privacy* (1997) pp 65-71; also available at `http://www.cis.upenn.edu/~waa/aegis.ps`; also “Secure and Reliable Bootstrap Architecture”, U.S. Patent No. 6,185,678, February 6th, 2001
17. L Lessig, *‘The Future of Ideas’*, Random House (2001)
18. P Samuelson, S Scotchmer, “The Law and Economics of Reverse Engineering”, Yale Law Journal, May 2002, 1575-1663, available at `http://socrates.berkeley.edu/~scotch/re.pdf`
19. C Shapiro, “Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard Setting”, in *Innovation Policy and the Economy*, MIT Press 2001; available at `http://faculty.haas.berkeley.edu/shapiro/thicket.pdf`
20. C Shapiro, “Antitrust Limits to Patent Settlements”. available at `http://faculty.haas.berkeley.edu/shapiro/settle.pdf`
21. A Gawer, MA Cusumano, *‘Platform leadership’*, Harvard Business School Press (2002)

Appendix

The following exposition is taken from [7], and uses an argument familiar to students of statistical mechanics. If there are $N(t)$ bugs left after t tests, let the probability that a test fails be $E(t)$, where a test failure counts double if it is caused by two separate bugs. Assume that no bugs are reintroduced, so that bugs are removed as fast as they are discovered. That is:

$$dN = -Edt \tag{1}$$

By analogy with theormodynamics, define a ‘temperature’ $T = 1/t$ and ‘entropy’ $S = \int dE/T$. Thus $S = \int tdE = Et - \int Edt$. This can be solved by substituting equation 1, giving $S = N + Et$. The entropy S is a decreasing function of t (since $dS/dt = tdE/dt$ and $dE/dt < 0$). So both S and N are bounded by their initial value N_0 (the number of bugs initially present) and the quantity $S - N = Et$ is bounded by a constant k (with $k < N_0$), that is:

$$E \leq k/t \tag{2}$$

Et vanishes at $t = 0$ and $t = W_0$, where W_0 is the number of input states the program can process. It has a maximum value $Et = k$. We now wish to show that this maximum is attained over a wide range of values of t, and indeed that $Et \approx k$ for $N_0 \ll t \ll W_0$. This will be the region of interest in most real world systems.

We can write the above equation as $Et = k - g(t)$ where $0 \leq g(t) \leq k$. Since $g(t)$ is bounded, we cannot have $g(t) \sim t^x$ for $x > 0$. On the other hand, if $g(t) = At^{-1}$, then this makes a contribution to N of $-\int g(t)dt/t = A/t$, which is reduced to only one bug after A tests, and this can be ignored as $A < k$. Indeed, we can ignore $g(t) = At^{-x}$ unless x is very small. Finally, if $g(t)$ varies slowly with t, such as $g(t) = At^{-x}$ for small x, then it can be treated as a constant in the region of interest, namely $N_0 \ll t \ll W_0$. In this region, we can subsume the constant and near-constant terms of $g(t)$ into k and disregard the rest, giving:

$$E \approx k/t \tag{3}$$

Thus the mean time to failure is $1/E \approx t/k$ in units where each test takes one unit of time.

More precisely, we can consider the distribution of defects. Let there be $\rho(\epsilon)d\epsilon$ bugs initially with failure rates in ϵ to $\epsilon + d\epsilon$. Their number will decay exponentially with characteristic time $1/\epsilon$, so that $E = \int \epsilon\rho(\epsilon)e^{-\epsilon t}d\epsilon \approx k/t$. The solution to this equation in the region of interest is

$$\rho(\epsilon) \approx k/\epsilon \tag{4}$$

This solution is valid for $N_0 \ll 1/\epsilon \ll W_0$, and is the distribution that will be measured by experiment. It differs from the ab-initio distribution because some defects will already have been eliminated from a well tested program (those in energy bands with $\rho(\epsilon) \sim \epsilon^x$ for $x > -1$), and other defects are of such low energy that they will almost never come to light in practical situations (those in energy bands with $\rho(\epsilon) \sim \epsilon^x$ for $x < -1$).