CHAPTER 3

A Theory of Disclosure for Security and Competitive Reasons: Open Source, Proprietary Software, and Government Systems

*Peter P. Swire**

A. INTRODUCTION

A previous article, "A Model for When Disclosure Helps Security: What is Different about Computer and Network Security?" proposed a model for when disclosure helps or hurts security and provided reasons why computer security is often different in this respect than physical security.[1] This chapter provides a general approach for describing the incentives of actors to disclose information about their software or systems. A chief point of this chapter is that the incentives of disclosure depend on two largely independent assessments: (i) the degree to which disclosure helps or hurts security,

---

* This chapter draws on discussions with numerous people over the past five years. I appreciate comments I received during the drafting of the chapter from Jon Callas, Whitfield Diffie, William Kovacic, David Ladd, Mark Lemley, David McGowan, and Adam Shostack, participants at the 2005 IPIL/Houston Santa Fe Conference "Transactions, Information and Emerging Law," the ACM Conference on Computer and Communications Security, and George Washington Law School IP Workshop. Chris Palmer of the Electronic Frontier Foundation has been an especially thorough and helpful commentator on the project.

1 Peter P. Swire, "A Model for When Disclosure Helps Security: What is Different about Computer and Network Security?" (2004) 3 J. on Telecomm. & High Tech. L. 163 ["Security Model"], online: http://ssrn.com/abstract=531782. A slightly updated version of the material was published as Peter P. Swire, "A Model for When Disclosure Helps Security: What is Different about Computer and Network Security?" in Mark F. Grady & Francesco Parisi, eds., *The Law and Economics of Cybersecurity* (New York: Cambridge University Press, 2006) 29.

and (ii) the degree to which disclosure creates competitive advantages or disadvantages for the organization.

Table 1 presents a 2 × 3 matrix in which disclosure for security and competition are assessed for three types of systems or software: open source; proprietary software; and government systems. The matrix indicates that there is a greater convergence on disclosure between open source and proprietary software than most commentators have believed. For instance, open source security experts use secrecy in "stealth firewalls" and in other ways. Open-source programmers also often rely on gaps in open source licences to gain a competitive advantage by keeping key information secret. Meanwhile, proprietary software often uses more disclosure than assumed. For security reasons, large purchasers and market forces often lead to disclosure about proprietary software. For competitive reasons, proprietary software companies often disclose a great deal when seeking to become the standard in an area or for other reasons.

Table 1

	Security	**Competition**
Open Source	Ideologically open; some "secret sauce" (Case 1)	Ideologically open; significant use of secrecy in practice (Case 2)
Proprietary Software	Monopolist on source code; disclosure based on monopsony and market structure (Case 3)	Monopolist on source code; disclosure based on how open-standards help profits (Case 4)
Government Systems	Information sharing dilemma (help attackers & defenders); public choice model & under-disclosure (Case 5)	Turf maximization & under-disclosure, e.g., FBI vs. local police for the credit (Case 6)

Despite this greater-than-expected convergence of practice for open source and proprietary software, there are strong reasons to believe that less-than-optimal disclosure happens for government systems. The tradition of military secrecy and the concern about tipping off attackers leads to a culture of secrecy for government security. Market mechanisms to force disclosure are less likely to occur for government agencies than for private companies. Competition for turf, such as the FBI's reputation for not sharing with local law enforcement, further reduces agency incentives to share information about vulnerabilities.

Part B of this chapter briefly recaps the relevant portions of the "Security Disclosure Model" I proposed in "Security Model." Part C shows the incen-

tive problems that exist when large databases are breached and the personal data of individuals is leaked. This sort of breach appears to be accompanied by significant externalities, so breach-notification statutes or similar measures are likely appropriate. Part D looks at the six parts of the matrix, analyzing the incentives for disclosure or secrecy for security reasons and competitive reasons, for open source software, proprietary software, and government systems.

This research provides a general approach for determining when disclosure is efficient for society and for describing the incentives actors face in disclosure or non-disclosure. The actual decision of whether to disclose in a given instance will depend on assessment of the empirical magnitude of the factors set forth in this chapter. The research provides, however, the first theoretical structure for assessing the issues. This theory is important to the design of systems and software in our information-rich age.

B. THE MODEL FOR WHEN DISCLOSURE HELPS SECURITY

"Security Model" presented the model for when disclosure of vulnerabilities helps security. This part briefly summarizes "Security Model," with emphasis on the aspects that are relevant to the current chapter. In brief, "Security Model" analyzed when disclosure would be socially optimal for security, taking into account the costs and benefits to all the parties involved.[2] This chapter continues the investigation of disclosure, looking at the incentives facing the key actors — those who design software or operate systems. In some settings, the incentives for key actors may lead to a large divergence between the socially optimal disclosure and the amount of disclosure actually made.

1) The Security Disclosure Model[3]

The Security Disclosure Model begins with a paradox. Most experts in computer and network security are familiar with the slogan, "there is no se-

---

2 More specifically, the Security Disclosure Model examined when disclosure would help "security," which is defined as "preventing the attacker from gaining control of a physical installation or computer system." "Security Model," *ibid.* at 205. The Security Disclosure Model does not explicitly address other important goals, such as disclosure to promote accountability (e.g., the *Freedom of Information Act*) or preventing disclosure in order to protect individual privacy. Important goals such as accountability and privacy should be considered in any overall decisions about the best levels of disclosure. *Ibid.* at 166–67 and 205.

3 "Security Model," *ibid.* at 167–75.

curity through obscurity."[4] For proponents of open source software, revealing the details of the system will actually tend to improve security, notably due to peer review. On this view, trying to hide the details of the system will tend to harm security because attackers will learn about vulnerabilities, but defenders will not know where to patch the vulnerabilities. In sharp contrast, a famous World War Two slogan warns "loose lips sink ships."[5] Most experts in the military and intelligence areas believe that secrecy is a critical tool for maintaining security. Both views — that disclosure helps security and hurts security — cannot be simultaneously correct, therefore we have a paradox. The task of "Security Model" was to explain the conditions for when each view, the open source view and the military view, is correct.

The first step toward resolving the paradox is to examine the effects of disclosure on attackers and defenders. Where disclosure on balance helps the attackers, then those defending the systems should rationally keep secrets. Where disclosure on balance helps the defenders, then disclosure should result.

By focusing on "effects on attackers"[6] and "effects on defenders,"[7] the Security Disclosure Model highlights the conditions under which the open source and "military" views are each correct. For open source, the usual assumption is that disclosure will not help attackers much or at all in a world of rapid communications among attackers, where exploits are rapidly learned by others. For open source, the next assumption is that disclosure of a flaw will prompt other programmers to improve the design of defences. In addition, disclosure will prompt many third parties — all of those using the software or the system — to install patches, or to otherwise protect themselves against the newly announced vulnerability. In sum, disclosure does not help attackers much but is highly valuable to the defenders who create new code and install it.

In contrast, the military assumptions highlight the ways that disclosure will assist the attackers. For a military base, for instance, the precise location of machine guns and other defences is closely guarded. A major goal is to hide the defences until it is too late for attackers, and they fall into traps.

---

4 *Ibid.* at 165 note 2.

5 *Ibid.* at 165 note 4.

6 See *ibid.* at 165–66. ("The first variable is the extent to which disclosure is likely to help the attackers, by tipping off a vulnerability the attackers would otherwise not have seen.")

7 See *ibid.* at 166. ("The second variable is the extent to which the disclosure is likely to improve the defence.")

In terms of disclosure helping defenders, the military traditionally uses its chain of command to tell fellow defenders what they need to know. There is no general broadcast of security flaws because such a broadcast would help the attackers but provide little or no information to fellow defenders.[8]

An important intermediate case for the present chapter is what I call the "information sharing paradigm," such as when the FBI or CIA is considering whether to share with officials in other agencies. To the extent the information is shared with the "good guys," then there can be strong assistance to the defenders because they might catch the terrorists before the attack occurs. To the extent the information is shared with the "bad guys," however, the information sharing can be the tip off that lets the attackers escape or change their plans. This dual effect of information sharing, to help defenders *and* attackers, is a helpful way to understand why it has been so important and yet so difficult a topic since the attacks of September 11, 2001.[9]

A fourth case concerns situations where additional disclosure about vulnerability will have small effects on attackers and defenders. An example is information in the public domain, such as detailed maps of Manhattan or Washington, DC. With maps so readily available in an online age, disclosure of one more map will make little difference to the risks of an attack against those cities.

---

8 In a workshop on this chapter, John Duffy made an excellent point: The "help-the-attacker" and "help-the-defender" effects can be described more generally as the prices that attackers and defenders face in gathering information about vulnerabilities. Widespread disclosure reduces information costs, such as when a software company publicly releases a patch to help defenders (i.e., users of software). Defenders still face the cost, however, of learning about the patch and deciding whether and how to implement it, with the result of substantially less than full implementation even of "free" patches.

9 One forum for extensive study of information sharing has been the Markle Foundation Task Force on National Security in the Information Age ("Task Force"). See Markle Foundation, "Task Force on National Security in the Information Age," online: www.markle.org/markle_programs/policy_for_a_networked_society/national_security/projects/taskforce_national_security.php. I was named an Associate to the Task Force in early 2005. My experiences there have confirmed my views that information sharing is often helpful but that poorly implemented information sharing also poses serious security and privacy risks. For my discussion of "The Bush Doctrine of Information Sharing," see Peter P. Swire, "America Faces the World on Privacy Four Years After 9/11" (Keynote Address at the Edinburgh Privacy Conference, 5 September 2005) [unpublished], online: www.peterswire.net/edinburgh_0905.ppt. I am currently writing a law review version of the topic for a symposium edition of the Villanova Law Review.

Table 2 pulls the four scenarios together. Notably, the open source scenario shows reasons for openness, with disclosure having a large "help-the-defenders" effect and a low "help-the-attackers" effect. The military scenario shows the opposite, with disclosure harming the defenders and helping the attackers. For information sharing, disclosure helps both the attackers and the defenders, making it unclear when to disclose. For the public domain, additional disclosure has minor effects.

Table 2

		"Help-the-Attackers" Effect	
"Help-the-Defenders" Effect		**Low**	**High**
	High	Open Source	Information Sharing
	Low	Public Domain	Military/Intelligence
Greater Disclosure Up and to the Left *Greater Secrecy Down and to the Right*			

2) Why Computer and Network Security Often Varies from Other Security[10]

Table 2 simplifies reality by asserting that open source situations should lead to disclosure and military situations should lead to secrecy. A chief goal of Table 2 is to organize the reader's thinking into when there is "no security through obscurity" or when instead "loose lips sink ships." Whatever the reader's prior assumptions about the desirability of disclosure, the 2 × 2 matrix shows that disclosure in some situations will help security (the open source scenario) and in other situations will hurt security (the military/intelligence scenario).

The next task in "Security Model" was to explain the conditions under which each scenario is likely to exist. For instance, one might believe that disclosure will help in computer security situations (which may use open source software), but hurt in military and other physical security situations.

10 This section is based on Parts II and III of "Security Model." See, generally, "Security Model," above note 1 at 175–207.

"Security Model" explained why there is no *logical* or *necessary* difference between cyber security and physical security.[11] Nonetheless, there are important reasons why there are *commonly* differences between the two. The organizing concept for when hiddenness helps security is that a hidden feature is more likely to be effective against the first attack, but less likely to be effective against repeated attacks. For example, imagine a path up to a fort that has a pit covered with leaves, with a sharpened stick at the bottom. The first time an attacker comes up the path, he might fall into the pit. Even if it works the first time, though, later attackers will likely not "fall" for the same trick. In short, obscurity may work against the first attackers, but it will not work once the attackers learn to watch for the hidden pit.

"Security Model" explored in detail the factors that affect when an attack is very unique — much like a first-time attack — or instead has low uniqueness and is like a repeated attack in which learning occurs. The function for uniqueness (*U*), or the usefulness of hiddenness for the defence, is

$$U = f(E, N, L, C, A)$$

Under this terminology, "high uniqueness" refers to situations where hiddenness is effective, due to a combination of high values of initial effectiveness (*E*) and ability to alter the defence (*A*), and low values for the number of attacks (*N*), learning from previous attacks (*L*), and communication among attackers (*C*). "Low uniqueness" refers to situations where the values are reversed.

Using this approach, major categories of computer and network security turn out to have much lower uniqueness than typical physical attacks. For firewalls, mass-market software, and encryption, the analysis is similar. They are all subject to repeated attacks (high *N*), attackers learn from previous attacks whether they succeed or not (high *L*), and they can communicate the exploit to other attackers (high *C*). Even if a hidden defensive trick is effective against an initial attack (high *E*), the effects of repeated, low-cost attacks will generally overwhelm the usefulness of hidden defences. Defenders can issue patches and new versions of the software (high *A*), but repeated attacks will, once again, soon reveal any secrets. In short, the open source assumptions are often a good approximation of reality for these computer-security situations. Disclosure of vulnerabilities can enlist others to write improved code (increasing *A*) and encourage other users to patch their systems (increasing *A*). Along with this high help-the-defenders effect,

---

11 *Ibid.* at 175.

disclosure will often have a low help-the-attackers effect because communication among attackers is already effective.

The equation suggests a different outcome for physical attacks on a military installation. For many physical attacks, the first attack is crucial (very low *N*). People might die in the first attack, which is far different than the costs to an attacker of trying to hack a firewall or software package. Even if attackers get into the fort, they might not capture the entire thing (low *L*), and they might not be able to radio out to their comrades what they learn (low *C*). Under such circumstances, hidden defences can be quite useful against the first attack, and defenders can often alter the location and types of defences before a second or third attack (high *A*). In short, the military/intelligence assumptions seem to be a good approximation of reality. Disclosure about defences, perhaps by a spy, will help the attackers but provide little or no benefit to the defenders.

"Security Model," explained in greater detail using the variables just discussed, demonstrates some situations where secrecy will help, even for computer and network security.[12] Notable examples include encryption and other private keys and passwords; situations under which it is difficult to discover software flaws; and surveillance where attackers do not learn about the surveillance in the course of the attack.[13] Similarly, "Security Model" explained some situations in which disclosure will help even for military and other physical security.[14] Examples include disclosure for deterrence (which helps the defenders by reducing the likelihood of attack); material that is already in the public domain; and situations in which there are repeated attacks (high *N*) and disclosure will help defenders learn how to respond (increasing *A*).[15]

More complete explanations of these topics are available in "Security Model." For now, the hope is that the reader has a basic sense of the approach. "Security Model" sought to explain when disclosure would either aid or harm overall security. The current chapter focuses instead on the incentives of actors to make the best level of disclosure.

---

12 *Ibid.* at 186–87 and 190–93 (indicating instances in which disclosure will hurt the defender and help the attacker).

13 *Ibid.*

14 *Ibid.* at 206.

15 *Ibid.*

C. SECURITY BREACH NOTIFICATION TO INDIVIDUALS

The discussion of incentives to disclose security flaws begins with a topic that has recently been the subject of intense legislative activity: notification about security breaches to the individuals whose data has been compromised.[16]

1) Describing the Externality

The problem is straightforward to describe. The first party is the organization that holds personal information in a database. The second party is the attacker — the outside hacker or malicious insider who is trying to get the data. The third parties are those whose personal information is in the database. The personal information may include items that create risks of harm to the third parties if the data is leaked. For instance, disclosure of bank account numbers might allow theft from the accounts, and disclosure of social security numbers might let the second party or others perpetrate identity fraud.

What is the incentive of the first party (the organization holding the data) to disclose the information or prevent it from being disclosed? It is possible that the incentives of the first party are aligned with those of the third parties. This is true, for instance, when two conditions are met: (i) third parties become aware of leaks of their data from the database; and (ii) market or other mechanisms are effective in disciplining the first party for data leaks.[17]

If these conditions are not met, however, then the first party will likely have an incentive to under-invest in protecting the data of third parties. Consider the possibility that leaks of data are hard to trace back to the first

---

16 See, for example, US, Bill S. 751, *Notification of Risk to Personal Data Act*, 109th Cong., 2005, [*Notification Act*] (proposing a bill that would require disclosure of unauthorized acquisitions of such information).

17 Discipline on the first party might occur in the market if its reputation is harmed in the event of breaches. Discipline might also occur, for instance, through legal mechanisms. An example would be if the third parties could sue for losses caused by the data breach.

The difficulty of tracing data leaks has also been a longstanding argument in favour of legislation or other measures to protect the privacy of individual information against sale or other intentional disclosure by the first party. See, for example, Peter Swire, "Markets, Self-Regulation, and Government Enforcement in the Protection of Personal Information by the U.S. Department of Commerce" (1997), online: www.ntia.doc.gov/reports/privacy/selfreg1.htm (examining the "uses and limitations of self-regulation" regarding the protection of personal information).

party where the leak occurred. This possibility happens often for personal data. It is usually very difficult for an individual, for instance, to figure out which of the databases containing her social security number was the source of a leak. Once a leak occurs, the individual may eventually discover that a criminal has used her bank account or social security number. The source of the leak, however, very often cannot be traced.

Untraced leaks from a database thus create a classic externality. In environmental protection, the factory (the first party) faces the cost of protecting against leaks, but the actual harms are on the third parties who are downstream from water pollution or downwind from air pollution.[18] In such circumstances, the factory has an incentive to pollute more than is desired by society. Much of environmental law consists of efforts to get the first party to face accurate incentives, so that the private decision by the factory owner matches the socially optimal decision, which includes effects on third parties.

Similarly, for protection of information in the database, the data holder (the first party) faces the cost of protecting against leaks, but the actual harms are on the third parties who suffer from the leaks. The incentives of the first party are thus to permit more leaks — more disclosure — than is desirable.

2) What, if Anything, to Do About the Externality?

Given the likelihood of an externality, the next question is what measures, if any, to take in response. Description of the externality shows that the problem arises from lack of disclosure to third parties about data leaks by the first party. A tailored response to that problem would be to change incentives so that third parties learn of the leaks.

California was the first to take this approach when it enacted Senate Bill 1386 in 2002.[19] In the wake of large data leaks early in 2005, many other states moved to create breach notification statutes. As of December 2005, at least eighteen states have passed legislation,[20] and Congress is seriously considering a federal breach-notification standard.[21]

---

18 For one explanation of the basic theory of environmental externalities, see US, Congressional Budget Office, "Federalism and Environmental Protection: Case Studies for Drinking Water and Ground-Level Ozone" (November 1997) c. 1, online: www.cbo.gov/showdoc.cfm?index=250&sequence=2.

19 Cal. Civ. Code § 1798.82 (West Supp. 2005).

20 BNA Privacy Law Watch, online: www.bna.com/products/ip/pwdm.htm.

21 See US, Bill H.R. 3140, *Consumer Data Security and Notification Act of 2005*, 109th Cong., 2005 (a bill requiring "consumer reporting agencies, financial institutions,

The California statute essentially provides that individual notice shall be given to California residents "whose unencrypted personal information was, or is reasonably believed to have been, acquired by an unauthorized person."[22] The breach is to be disclosed in writing without unreasonable delay, with certain exceptions.[23] "Personal information" means a person's name in combination with any listed data element, such as social security number, driver's licence number, or "[a]ccount number, credit or debit card number, in combination with any required security code . . . that would permit access to an individual's financial account."[24]

In terms of the best approach for breach notification, I would like to make a few suggestions based on my experience with system owners, consumer groups, and regulators. The goal is to avoid both under-disclosure and over-disclosure. Under-disclosure quite likely exists in the absence of a statute (because of the incentives for under-disclosure described above). Over-disclosure could also be a problem. For first parties, there would be the expenses of first-class postage to very large classes, perhaps even for small and everyday levels of security flaws. For consumers, a blizzard of disclosures would swamp the signal in the noise. That is, consumers who received numerous notices would not know how to respond to the subset that posed a serious threat. Individual consumers would generally prefer to get notice when there are reasonable measures they should take in response.

To avoid both under- and over-disclosure, it makes sense to look at a security breach as a systemic issue over time. First, there should likely be a sunset on breach-notification statutes. We are at the early stages of knowing how to implement such statutes, and the sunset would likely spur better re-examination of the issue over time. Second, we should seriously consider a two-step regime. For more serious breaches, there would indeed be first-class mail notice to individuals, especially where there are concrete steps the individuals should take in response to the risks. For less serious breaches,

---

and other entities to notify consumers of data security breaches involving sensitive consumer information"); *Notification Act*, above note 16, (a bill requiring "Federal agencies, and persons engaged in interstate commerce, in possession of data containing personal information, to disclose any unauthorized acquisition of such information").

22 Above note 19, § 1798.82(a).

23 *Ibid*., §§1798.82(a), (c), and (g). Email notices are only allowed if they comply with federal law regarding electronic signatures. § 1798.92(g)(2). In our current world of spam and phishing attacks, email notices might easily be filtered out or ignored by the recipient.

24 Above note 19, §1798.82(e).

which do not merit this individualized notice, there should be mandatory reporting to some agency such as the Federal Trade Commission. This sort of reporting would serve two major goals. It would mean that significant breaches, which do not merit notice to individuals, would be subject to action by the database holder. In this way, significant-but-not-major breaches would be addressed by data holders. In addition, it would allow the Federal Trade Commission to accumulate data about breaches, preparing the way for better re-examination of breach notification statutes over time.

More can be said about the best system for handling security-breach notification. For purposes of this chapter, the recent legislative activity shows the importance of addressing the incentives of organizations to provide the proper level of disclosure about security issues.

D. INCENTIVES FOR DISCLOSURE AND SECRECY FOR SECURITY AND COMPETITIVE REASONS

Part C of this chapter, on breach notification, concerned disclosure of information about third parties (customers) for data held by a first party (the system owner). The key point of this discussion was to show a potentially significant externality — that the system owner would not expend resources to protect third parties against harm from release of their data.

This part engages in a wider inquiry into incentives of system owners and software writers ("first parties") to disclose information or keep it secret. A new concept here is that system owners have a distinct calculus based on two different motives, the "security motive" and the "competitive motive." The security motive concerns the incentive of the first party to disclose or not, based on achievement of the security goals of the first party and other parties. Notably, what is the rational calculus for when disclosure will help ("there is no security through obscurity") or when instead secrecy will lead to better security ("loose lips sink ships")?[25] The competitive motive concerns the incentive of the first party to win in the marketplace against its competition. Notably, the first party will seek to determine when greater secrecy will help it competitively, such as with trade secrets, or when, instead, greater openness will enhance competitiveness, such as when use of an open standard attracts more business for a software writer.

---

25 As in "Security Model," security is defined as "preventing the attacker from gaining control of a physical installation or computer system." "Security Model," above note 1 at 205.

The two motives are generally analyzed separately in this chapter. In many instances, the security motive will not have a large effect on a company's bottom line — the question is whether enhanced security or secrecy is more effective at protecting security. In other instances, the security motive does significantly affect the bottom line, and analysis of the security motive proceeds here based on what helps the security of the affected parties. By contrast, the competitive motive includes all the incentives that a company faces that are not based on achieving security for itself or for other relevant parties. To the extent that disclosure for security purposes affects a company's bottom line, then analysis of the security motive will affect the rational degree of disclosure for competitive purposes.

The analysis here focuses on three categories of actors: (i) open-source-software writers, (ii) proprietary software writers, and (iii) government agencies. The three categories generate quite different and interesting results. In addition, the three categories map the terrain: private actors with a presumption of disclosure (open source); private actors with a presumption of secrecy (proprietary software where source code is not revealed); and public-sector actors.

Table 1 summarizes the more detailed discussion below. The table shows the key findings for open source, proprietary, and government actors, for both the security and competitive motives.

1) Case One: Security Incentives and Open Source Software

The first case examines, from a security standpoint, the incentives to disclose or not for designers and users of open source software.[26] Here is where the maxim that "there is no security through obscurity" has its greatest support. Proponents of open source, including the GNU General Public License, are proud that there is far greater disclosure than for proprietary software:

> The twist in the principal open-source model — including the General Public License ("GPL"), among the oldest and best-known public licence — is that if the software is distributed to third parties, it MUST be distributed using the open source licensing model, making the model "self-perpetuating." Thus, instead of guarding the "secret sauce," the

26 For a discussion of the legal, technical, and developmental differences between open source and proprietary software, see Jonathan Zittrain, "Normative Principles for Evaluating Free and Proprietary Software" (2004) 71 U. Chicago. L. Rev. 265 at 268–73.

> GPL-Open Source approach not only makes it available but also mandates that all "secret sauce improvements" are made available in the same manner.[27]

Even here, however, there turns out to be unsuspected areas where hiddenness — the use of "secret sauce" — is used for security purposes.

a) When hiddenness can help open source security

As the previous quote indicated, any use of hiddenness for open source software seems inherently contradictory. After all, the GNU General Public License, Version 2.0, says to those who distribute software under that licence: "[Y]ou must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code."[28] The heart of the open source requirement, as the name implies, is that the source code will be open. It may seem, thus, that there can be no secrecy.

Nonetheless, there are at least three respects in which secrecy can be and is used to help security for open source software. The first, password and encryption key secrecy, is entirely uncontroversial and was discussed in detail in "Security Model." Even encryption experts, who argue most vehemently against "security through obscurity," agree that passwords and encryption keys should remain secret.[29] It is good common sense that revealing one's passwords will help the attackers but not provide any benefit to defenders. Modern encryption systems are designed, in fact, to have the clearest possible separation between the password or key (kept secret) and the rest of the cryptosystem (kept open so that other experts can test for flaws).[30] In short, a secret key or password is essential to use the system, but there is no benefit to allowing outsiders to have the key.

---

27 Randall M. Whitmeyer, "Open Source Legal Issues and Controversies" (7 August 2004), online: www.techjournalsouth.com/news/article.html?item_id=649.

28 "GNU General Public License," online: www.gnu.org/copyleft/gpl.html ["GNU General Public License"]. For an extensive set of links about the GNU Public Licence, see Groklaw, "GPL Resources," online: www.groklaw.net/staticpages/index.php?page=20050131065655645.

29 "Security Model," above note 1 at 190–91.

30 Interview with Whitfield Diffie, Vice President, Sun Fellow & Chief Security Officer, Sun Microsystems (9 September 2005). Research for this project included an enlightening conversation with Whitfield Diffie, one of the authors of the foundational Diffie-Hellman algorithm for public key encryption. Diffie observed that the "security through obscurity" debate may have been led somewhat astray because techniques for separating the key from the cryptosystem were well developed for encryption but much less developed (and perhaps not applicable) for other security

The second area of potential secrecy involves surveillance by the defenders. As discussed in "Security Model,"[31] disclosure about surveillance, and especially its sources and methods, often helps the attackers more than the defenders. With surveillance, there is typically low learning by attackers (L), because the surveillance is designed specifically to make it difficult for attackers to detect. Because of this low L, it is often rational for defenders to keep surveillance techniques secret. (Also as described in "Security Model," secret surveillance may or may not be ultimately desirable, based on concerns including privacy, accountability, and long-run improvement of the security of systems.[32])

One common type of surveillance by defenders is an intrusion detection system, which seeks to detect the existence and nature of intrusions by attackers.[33] Intrusion detection software is available in both proprietary and open source forms.[34] Logically, there appear to be advantages to keeping some aspects of intrusion detection software secret, much as one would try to keep secret the placement of hidden cameras that watch for burglars.

Secrecy is even more important, though, for the aspect of intrusion detection software known as "honeypots," which "emulate real running operating systems to serve as a bait for potential attackers."[35] The idea of a honeypot is that the intruder is attracted to the honey — the apparently sweet target. The honeypot is under surveillance, however, and the defenders thereby learn about the type and number of attacks. Honeypots are available in open source code, and are used regularly by open-source programmers,[36] yet they rely on secrecy. Having a honeypot obviously will not work if the intruder knows that it is a fake. The use of honeypots illustrates how surveillance relies on secrecy, even for open source systems.

---

problems. For situations in which it is not feasible to separate the key from the rest of the security system, the clear distinction between hidden elements (the key) and public elements (the system) would not be sustainable.

31 "Security Model," above note 1 at 191–93.

32 *Ibid.* at 193.

33 For a detailed set of questions and answers about intrusion detection, see: SANS Institute, "Intrusion Detection FAQ," online: www.sans.org/resources/idfaq ["Intrusion Detection FAQ"]. SANS Institute defines "intrusion detection" as "the art of detecting inappropriate, incorrect, or anomalous activity."

34 Snort is one well-known intrusion detection system that is open source. See "About Snort," online: www.snort.org/about_snort.

35 Alexander Prohorenko, "Open Source Intrusion Detection: No-cost System Lockdown" (9 November 2004), online: www.devx.com/security/Article/22442.

36 *Ibid.*

Along with passwords and surveillance, the third area of potential secrecy is anything outside the scope of an open source licence, such as the GNU General Public License or others approved by the Open Source Initiative.[37] Even for the GNU Public License, the most widely used, there appear to be significant ways that programmers can use secrecy. One notable way is for programmers to use non-standard configurations and settings for the system. The term "configurations" is used in computer systems essentially to refer to the choice of software and other components in ways that affect system function.[38] The term "settings" has been defined as "[p]arameters of a system or operation that can be selected by the user."[39] The terms "configurations" and "settings" do not appear in the GNU General Public License, but the text of the licence seems to permit users to create or alter configurations and settings.[40]

The importance of non-standard configurations and settings was highlighted to me by an experienced open source programmer, Jon Callas, who said: "I'm not afraid to use a little secret sauce."[41] Callas gives the example of

---

37 It is not my intention here to enter the debate about what should be considered a true "open source licence." At the time of this writing, the Open Source Initiative has recognized over fifty licences as qualifying for its service mark. For a list of licences, helpful definitions, and links about the relevant terms, see "Open Source Licence," online: http://en.wikipedia.org/wiki/Open-source_license.

38 See National Communications System Technology & Standards Division, *Telecommunications: Glossary of Telecommunication Terms* (1996), online: www.tiaonline.org/market_intelligence/glossary/index.cfm?term=%26%23TOZRR%23M%0A (Configuration is, "[i]n a communications or computer system, an arrangement of functional units according to their nature, number, and chief characteristics . . . . Configuration pertains to hardware, software, firmware, and documentation.").

39 Glossary — Networking Terms Related to Remote Scope, online: www.micro2000.co.uk/products/remotescope/glossary.htm.

40 See "GNU General Public License," above note 28. For instance, paragraph 0 of the licence states: "Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted . . . ."

41 Email from Jon Callas, Founder, Chief Technical Officer & Chief Security Officer, PGP Corp., to author (20 September 2005) (on file with author). The term "secret sauce" may trace back to the "special sauce" in a McDonald's ad that played often during the childhood of many current-day programmers: "Two all beef patties, special sauce, lettuce, cheese, pickles, onions on a sesame seed bun." Readers who can illuminate the history of the terms "secret sauce" or "special sauce" in programming are welcome to contact the author.

secret sauce he implemented for Secure Socket Layers (SSL), a cryptographic protocol that provides secure communications on the Internet. He says:

> In the generic case of SSL, we want secure interoperation of arbitrary web browsers and web servers. In the case I mentioned where I did some tweaks to SSL's security I didn't want interoperability, I only want *my* systems to interoperate, so there is usefulness in combining nonstandard configurations with a standard architecture.[42]

Using unknown and non-standard configurations and settings can have significant advantages to system owners. Notably, this use of obscurity can frustrate standardized attacks by "script kiddies" or other inexpert hackers.[43] Even for more skilled attackers, it may be rational to shift to an easier target when a target initially resists an attack.[44] It may thus be rational for a defender to use hidden and non-standard configurations and settings even where they would not withstand a determined attack. In terms of the model for when disclosure helps security, the non-standard configurations and settings increase the uniqueness — they reduce N (the number of attacks against a particular defence) — and make it more like a first-time attack. In such circumstances, hiddenness is more likely to be effective. In the words of Jon Callas: "Obscurity is camouflage, security is armor."[45] Either can be useful, depending on the circumstances. They can also be useful when working together, much like tanks that are often camouflaged.[46]

A second current example of secrecy in open source software, the so-called "stealth firewall,"[47] fits the idea of camouflage. A standard way to

---

42 *Ibid.*

43 For discussion of script kiddies, who are unskilled programmers who merely follow a script rather than understanding how to write code themselves, see Swire, "Security Model," above note 1 at 180 note 21.

44 See *ibid.* at 189 (discussing how relatively strong security may shift an attack to a different target).

45 Email from Jon Callas, above note 41.

46 See "Tank Research and Development," online: http://en.wikipedia.org/wiki/Tank_research_and_development. Indeed, some experts emphasize the need for stealth technologies, believing that "the future of tanks lies in invisibility rather than invincibility." For an essay on tank camouflage when tanks were introduced in World War I, see Patrick Wright, "Cubist Slugs" (2005) 27(12) London Review of Books, online: www.lrb.co.uk/v27/n12/wrig01_.html.

47 The discussion here of stealth firewalls is based on (i) Interview with Chris Palmer (14 April 2005); and (ii) Michael Shinn & Scott R. Shinn, "Stealth Firewalling with Linux," *Linuxworld Magazine* (27 February 2005), online: http://linux.sys-con.com/read/48126.htm.

attack a firewall is to scan it, checking its existence and behaviour. A nice attribute of a stealth firewall is that it does not have to have an IP address that is scannable. The stealth firewall thus resists a common attack.

In this example, the source code for the stealth firewall may well be open source. The defence, however, depends on hiddenness from observation by the attacker. More generally, "Security Model" described the usefulness of hiddenness where the attacker has low learning (L) from attacks.[48] Similar to stealth firewalls, intrusion detection systems and other surveillance by the defender are more effective if attackers cannot tell how they operate.[49] Once again, the low level of L from prior attacks means that hiddenness is more likely to be effective, even against repeated attacks. Disclosure of the details of a stealth firewall or an intrusion detection system helps the attackers, but offers little likelihood of helping the defenders.

b) Implications of hiddenness in open source security

The discussion here highlights the incentives for open source programmers to use secrecy to produce a low N, such as through non-standard configurations and settings, or a low L, such as through stealth firewalls and hidden intrusion detection systems. These strategies will often be permitted under open source licences because the secrecy is not at the level of source code that must be disclosed under the licence.

These reasons to use secrecy have not been highlighted in the open source literature to date. Some open source proponents may be uncomfortable using the secrecy strategies discussed here, believing they smack too much of "security through obscurity." However, the examples used here came directly from interviews over time with open source programmers, who responded to the question: "Where, if anywhere, is there secrecy to promote the security of open source systems?" Within the overall paradigm of open source software, which so emphatically emphasizes the security advantages of disclosure, there are situations where secrecy appears to improve security. Those situations, as described here, fit well into the model of disclosure presented in "Security Model."

As a policy matter, the next question is whether there are externalities or other reasons to be concerned about harm from secrecy as used by open source programmers. The answer appears to be no. The uses of "secret sauce," such as stealth firewalls or non-standard configurations, affect the

---

48 See "Security Model," above note 1 at 176.

49 The role of secrecy for effective surveillance is discussed *ibid.* at 191–93. See also "Intrusion Detection FAQ," above note 33.

system owners themselves. There does not appear to be any significant externality where the failure to disclose leads to harm to others. In contrast to the large externality, discussed above, for disclosures of security breaches, the analysis here does not support any call for legislation or other measures with respect to disclosure for open source systems.

2) Case Two: Competitive Incentives and Open Source Software

For open source programming today, the use of secrecy is likely even more prevalent for competitive reasons than for the security reasons just discussed. My discussion in this section is deliberately somewhat provocative — the emergence of large firms in open source software and large corporate-users of open source software has changed practices considerably in the last decade. My prediction is that secrecy for competitive reasons will become an increasingly evident part of the open source world in the coming years.

a) The openness of open source software

As with the security discussion, it is important to begin with an understanding about the baseline of open source disclosure before examining ways that hiddenness also exists.

Open source software eschews key legal and technological measures that are used by proprietary software companies to protect their work. Most fundamentally, open source licences such as the GNU General Public License require disclosure of the source code.[50] For those who wish to reverse engineer a proprietary program, it is often expensive and difficult to do so and get usable source code. The closed or hidden nature of proprietary code often creates a large obstacle for those who wish to compete with the proprietary company. By contrast, the readability of open source code means that a competent coder may be able to quickly equal or surpass the insights of the person who wrote the original code.

Open source software not only lacks technical protection against competition and disclosure, but it lacks traditional legal protections. The "copyleft" nature of the GNU General Public License means that a later coder is forbidden from getting a copyright on material derived from licensed code.[51] If the open source movement is successful in ongoing patent disputes

50 "Open Source Initiative," online: www.opensource.org/docs/definition.php.

51 "GNU General Public License," above note 28.

such as the SCO litigation,[52] then patents will not be available to protect, at least, most open source code, nor, according to one expert in the field, will trade secret protection be available: "[T]rade secret protection is singularly inapplicable to open source software. The accessible and open source code would almost always defeat the trade secret status by disclosing the secret."[53]

The commitment to openness in the open source movement goes beyond these technological and legal commitments to openness. The ideology of free software, developed most prominently by Richard Stallman and the Free Software Foundation,[54] has had a powerful influence on a vast community of programmers. The view that free software is morally superior to proprietary software is accompanied, in turn, by influential academic commentary that provides theoretical explanations for how the open source movement initially developed and why it seems especially well-suited to creating software in a networked setting.

Yale Professor Yochai Benkler has offered a clearly defined theory for situations in which open source collaboration will have particular advantages over a proprietary approach: "[T]here will be conditions under which a project that can organize itself to offer social-psychological rewards removed from monetary rewards will attract certain people, or at least certain chunks of people's days, that monetary rewards would not."[55] Benkler emphasizes three conditions. First, the project must be modular, so that many people can contribute incrementally over time. Second, the project should have the right level of lumpiness, so that individuals can make contributions of a manageable size. Third, "a successful peer production enterprise must have low-cost integration, which includes both quality control over the modules and a mechanism for integrating the contributions into the finished product."[56] The peer-review aspect of open source production can

---

52 The SCO Group has sued IBM and other companies, with a principal allegation being infringement of patents that it says applies to code contributed to open source projects. See Amended Complaint, *SCO Group, Inc. v. IBM* (16 June 2003), Utah, No. 03-CV-0294 (Dist. Ct.), online: www.sco.com/scoip/lawsuits/ibm/ibm-25.pdf.

53 Greg R. Vetter, "The Collaborative Integrity of Open-Source Software" (2004) Utah L. Rev. 563 at 588.

54 Richard Stallman is the President and Founder of The Free Software Foundation. Free Software Foundation, GNU's Who-GNU Project, online: www.gnu.org/people/people.html. For more information about the Free Software Movement or Stallman, see Free Software Foundation, online: www.fsf.org.

55 Yochai Benkler, "Coase's Penguin, or, Linux and The Nature of the Firm" (2002) 112 Yale L.J. 369 at 378 [Benkler].

56 *Ibid.* at 378–79.

serve this integration function. Where all three conditions exist, a peer-production process, based on openness, can succeed, even in the absence of substantial monetary awards.

b) Hiddenness and the competitive incentives of open source users

Before looking at the incentives that face those who create open source code — the coders, developers, programmers, and so on — it is useful first to examine the users of that code. As open source code becomes more pervasive in modern organizations, decisions about openness versus hiddenness will increasingly turn on the needs of these open source users, such as manufacturers, services companies, government agencies, and so on. These users have their own imperatives about what should be disclosed.

Take the example of the widget manufacturer who uses open source code for standard tasks, such as supply-chain management, inventory control, or product testing. Using expertise about the widget sector, suppose that persons working for the manufacturer find a way to cut costs by 5 percent. They write new code, for use inside the company, that implements the cost-cutting measures.

Next, suppose you are the chief information officer (CIO) for the manufacturer, and you meet with the company president to decide whether to disclose the new code. As CIO, you might try to explain the beliefs of the open source community, including the importance of giving back code for use by the broader community.[57] The president is likely to be unimpressed, and perhaps even incredulous: "You mean we should tell our competitors how to get the 5 percent savings, after we invested all that time and money to learn how to do it!?"

This story of the widget manufacturer can be generalized. Corporations routinely make decisions about what to disclose or keep secret for competitive purposes. Confidential business information is a broad category, with common examples including business strategies, research that is not fully protected by patents, know-how relevant to ongoing operations, and trade secrets. To the extent that release of code will tip off competitors or other-

57 For one influential discussion of the philosophy of the free software movement, see Eric S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, rev. ed. (Beijing: O'Reilly, 2001).

wise reduce profits, the rational corporate decision maker will decide to keep the code in-house.[58]

Open source proponents hope and expect that open source code will pervade business organizations in the future. If and as this occurs, public release of that code will increasingly provide clues to competitors about the activities of the company that used and modified the code. The incentives for open source users to keep their competitive secrets will then be in conflict with the norm of open source programmers to disclose the code.[59]

c) The puzzle of why corporations invest in producing open source software

Along with the incentives for secrecy for open source users, there is the well-recognized puzzle about why individuals and firms will rationally develop open source code. Benkler's theory coherently sets forth certain conditions under which social-psychological rewards can provide the motivation.[60] Later, he briefly examines what he calls "indirect appropriation mechanisms."[61] He writes, "These range from the amorphous category of reputation gains to much more mundane benefits, such as consulting contracts, customization services, and increases in human capital that are paid for by employers who can use the skills gained from participation in free software development in proprietary projects."[62]

The discussion here, by contrast, emphasizes that the appropriation mechanisms are often direct rather than indirect.[63] More surprisingly, *the*

---

58 Note that this discussion does not assume that the secrets would be protected against a determined adversary, such as one who resorted to industrial espionage. In business competition, it will often be rational not to disclose information if keeping the information secret simply raises costs to competitors or reduces the likelihood that they will learn the competitive secrets.

59 The extent to which the open source community can develop strategies to reduce such conflict with incentives for users to guard competitive secrets is a topic for further research. For instance, in some settings, source code might be released in ways that do not reveal corporate secrets. If readers are aware of such efforts to reconcile these conflicting norms of secrecy and disclosure, the author would be glad to learn of them.

60 See Benkler, above note 55 at 378–79.

61 *Ibid.* at 424.

62 *Ibid.* at 424–25 (footnote omitted).

63 As the draft of this chapter was being completed, Ronald Mann released an excellent working paper that addresses the puzzle of why corporations are investing large sums in open source software. Ronald J. Mann, "The Commercialization of Open Source Software: Do Property Rights Still Matter?" online: http://papers.ssrn.com/

economics of open source programming increasingly relies on (at least temporary) secrecy. The incentives facing those developing open source systems rely on secrecy in at least three distinct and substantial ways: staying ahead of the curve; systems integration and related services; and web services.

d) Hiddenness and the competitive incentives of open source developers

Significant incentives exist for secrecy on the part of open source developers in order to stay ahead of the curve, offer system integration and related services, and create open source code in ways that do not trigger disclosure obligations under the relevant licence.

i) Staying ahead of the curve

As I spoke to open source programmers about secrecy and openness, one who preferred not to be identified said, "I always try to stay six months ahead of the curve."[64] This programmer has regularly contributed code to various open source projects. He does not disclose, however, some aspects of his work that will give him a competitive advantage. A similar point has been made by Robert Lefkowitz, former director of Open Source Strategy at Merrill Lynch: "And then, there's the secret sauce for which you wish to

---

sol3/papers.cfm?abstract_id=802805. Professor Mann agrees with arguments made in this chapter that the financial incentives for investment are direct, as opposed to the indirect approach emphasized by Benkler, *ibid.* at 19–20. Professor Mann explains that much of open source investment follows the classic "value chain" model, which dictates that a company should "foster the commoditization of those portions of the stack in which the company does not have a core competency, so that it can earn high(er) returns for those portions of the stack in which it can defeat its competitors." *Ibid.* at 20. Essentially, the open source approach serves as a precommitment strategy, in which the major open source corporations cannot profit directly from sales of Linux or other open source software, but all of them become committed to an open source approach.

Professor Mann makes a number of points that are similar to those made here, including his description of how open source companies are able to profit from services. Nonetheless, it is not clear how well his precommitment strategy explains the large investments he documents. The company-specific returns emphasized in this chapter seem like a more straightforward way to explain the investments of hundreds of millions or billions of dollars by individual companies.

64 The programmer's reluctance to be named is a sign, perhaps, of the culture of openness that is prevalent in the open source community. Admitting that one is holding back information from the community can be a source of shame.

capture an innovation premium. You can take that open source stuff, add your secret sauce, and create some very nice commercial products."[65]

The idea of staying ahead of the curve is a powerful one. Although it is sometimes risky to be out front, there may be a substantial first-mover advantage in a particular market niche.[66] More generally, there are ongoing advantages to being somewhat ahead of the curve. As your competitors work to catch up to where you were at one point, you can seek to keep your lead through ongoing investment.

For writers of software, the payoff can come in various forms, such as through general reputation or positive reviews from satisfied clients. Applied to open source programmers, the idea of staying ahead of the curve suggests the business sense of what the anonymous programmer has done — release some code to the community to build reputation, while holding some valuable code back in order to provide premium service to clients.

The idea of staying ahead of the curve is essentially one of temporary secrecy. In a rapidly changing environment, getting the source code from a year or two ago will often not be very satisfying. Purchasers will be motivated to go to the suppliers of up-to-date solutions.

ii) Systems integration and services

Why have corporate giants such as IBM and many others invested so heavily in open source in recent years? The elegant peer-production model of Yochai Benkler is designed to explain why individual programmers will seek social-psychological rewards rather than monetary profit.[67] That model, by its own terms, does not apply to profit-maximizing corporations. The proposal here is that the corporations will differentially invest in ways that the competitive advantage does not need to be disclosed under the open source licences. A major category of corporate effort is in the area of systems integration and related services.

IBM employee Michael Nelson said it succinctly: "We take very valuable open source software, and build it into a suite of services that's even

---

65 "Open Source Gets Down to Business" *Technology Review* (25 July 2003), online: www.technologyreview.com/Infotech/13258/?a=f (interview with Robert Lefkowitz, former Director of Open Source Strategy, Merrill Lynch).

66 See Jonathan M. Barnett, "Private Protection of Patentable Goods" (2004) 25 Cardozo L. Rev. 1251 at 1257–66 (discussing generally the economics of first-mover advantage).

67 Benkler, above note 55 at 378.

more valuable."[68] The logic behind this bland statement is important. The economic incentive is that the innovation and comparative advantage happen elsewhere (i.e., not in the source code). Under the licence terms, the source code must be revealed upon distribution. The techniques of services and system integration do not. The importance of integration is echoed by *Computerworld* columnist Frank Hayes:

> Once, IT would have looked for unique advantage by writing big custom applications. But today that takes too long and is too inflexible. Instead, open source-using companies like Google and Yahoo have figured out that their secret sauce is in the way they put together pieces of IT—software, hardware, networks and practices. Anyone can acquire the gear these companies use. How they put it together is the difference.[69]

A company's comparative advantage in systems integration also responds directly to Professor Benkler's model. He postulated that "a successful peer production enterprise must have low-cost integration, which includes both quality control over the modules and a mechanism for integrating the contributions into the finished product."[70] There is an empirical question about when a peer-production model will be superior to a corporate-production model for these aspects of quality control and integration. It seems plausible, however, that the reputation of a large company for providing quality control and integration will often give it an advantage in the marketplace.[71] To the extent that an advantage does occur, the lucrative market for quality control and integration may shift to corporate players, and away from the peer-production process.[72]

---

68 Email from Michael Nelson, IBM Employee, to author (21 October 2005) (on file with author).

69 Frank Hayes, "Deliver the Goods" *Computerworld* (8 August 2005) 54 at 54, online: www.computerworld.com/softwaretopics/software/appdev/story/0,10801,103728,00.html.

70 Benkler, above note 55 at 379.

71 For a thorough examination of the factors that can give comparative advantages to firms to internalize activities, including quality control and reputation effects, see, generally, Oliver E. Williamson, *Markets and Hierarchies: Analysis and Antitrust Implications* (New York: Free Press, 1975).

72 A web posting made by a frustrated open source programmer offers insight into how such marketplace advantages look to someone competing with the big companies:

> IBM often brags about billions in profits from Open Source because of consulting revenues. The problem with that is the guy who contributed most to the

iii) Web services and other in-house use of open source

The current structure of the GNU General Public License creates another financial incentive for secrecy. As noted above, that licence requires publication of the source code upon distribution.[73] Before distribution occurs, however, there is no requirement to reveal the source code.

A famous example of a company adopting this approach is the online retailer Amazon.com (Amazon). Publishing impresario Tim O'Reilly has analyzed the subject: "Even though *amazon.com* is built on top of open source software, the licences compel no release of Amazon source because its software is never distributed. In the new world of the Internet, software is a service, not an artifact."[74]

The Amazon example illustrates the powerful business incentive that exists under the current GNU General Public License — open source code can be used, but need not be made public so long as the code is not "distributed" to others.[75] This "use-but-don't-disclose" aspect of the General Public License creates the possibility of supplying proprietary solutions over the Internet. Amazon uses its code, its "secret sauce," inside of Amazon. In this respect, Amazon acts in part as a developer of open source code — its employees and agents write code that may give Amazon a competitive advantage as an online retailer. Amazon also, in part, acts as a user of open source code. Like the widget manufacturer discussed above, Amazon may be reluctant to reveal proprietary information that is not about the code itself. In addition, companies that provide software as a service, but do not

---

> original source code who did it for the love of coding probably didn't see too many of those dollars.
>
> While it is possible for guy [*sic*] who did the coding to sell his consulting services, how does he compete with IBM global services? He doesn't have 1/100th the reputation, sales, and marketing force of IBM global services. IBM global services can just take your GPL code and implement it however they want. Once in a while, they might throw you a few crumbs for you [*sic*].

Posting of george_ou to TalkBack on ZDNet (14 June 2005), online: www.zdnet.com/5208-10535-0.html?forumID=1&threadID=11136&messageID=222450&start=-30.

73 See "GNU General Public License," above note 28 and accompanying text (providing that distributors of software under the General Public License must "give the recipients all the rights that [they] have").

74 O'Reilly Network, "Ask Tim: Amazon and Open Source" (February 2004), online: www.oreilly.com/pub/a/oreilly/ask_tim/2004/amazon_0204.html.

75 "GNU General Public License," above note 28.

trigger the General Public License by distribution through a CD or floppy disk are under no obligation to reveal their source code.[76]

iv) Implications of the analysis

The discussion here has highlighted important reasons why those involved with open source software will rationally rely on secrecy, at least to some extent. On the user side, the incentives to protect confidential business information will remain even for businesses that use open source software. For developers, there are incentives to use secrecy to stay ahead of the curve, to provide systems integration and related services, and to use open source software in ways that do not involve "distribution" and thus do not trigger obligations to disclose. One result has been an open source universe that is dominated by services, where profits can be captured by leading companies, rather than by the commodity software products that are so prominent in the proprietary software space.

In response to this possible shift toward secrecy, there has been discussion of a Version 3.0 of the GNU General Public License, which would address the disclosure issue. According to Sleepycat Software CEO Mike Olson, the next generation of the licence will address the distribution "loophole": "If you look at the market, Yahoo, eBay, IBM, Amazon, Google have all sunk millions into the GPL infrastructure."[77] When it comes to the distribution loophole, Olson added, "Not only are we changing the rules, we are changing them retroactively."[78]

The exact language for Version 3.0 is not known yet, and Eben Moglen, the head of the Software Freedom Law Center and a leader of the drafting process, has withheld comment until a discussion draft is released.[79] Nonetheless, it will be most interesting to see the extent to which major corporate players are willing to shift to Version 3.0 along the lines discussed by Mike Olson. The analysis in this chapter suggests that Version 3.0 will be a very tough sell. The business strategies of major corporations seem to have evolved to what is permitted under Version 2.0, with secrecy an important competitive advantage. If Version 3.0 looks the way Mike Olson has stated, then there may be a major moment of decision for open source developers: to what extent will the requirements for openness be extended, as Olson

---

76 Michael Singer, "Insider Hints at GPL Changes" (7 April 2005), online: www.internetnews.com/dev-news/article.php/3495981.

77 *Ibid.*

78 *Ibid.*

79 See *ibid.*

suggests, or to what extent instead will corporate business plans rely on the categories of secrecy described in this chapter?

3) Case Three: Security Incentives and Proprietary Software

The third case examines, from a security standpoint, the incentives to disclose or not for designers of proprietary software. The analysis here shows a surprisingly diverse set of factors that create incentives to disclose security vulnerabilities.

a) The hiddenness of proprietary software

Proprietary software traditionally uses both technological and legal methods to protect against unauthorized use by others. The technological method is to not reveal the source code. Outsiders who wish to reverse engineer the software thus need to engage in often elaborate efforts to figure out how to write code that will have the same function as the proprietary program.[80] The legal methods often include copyright on the program, and may also include patent and trade-secret protection. Both the technical and legal mechanisms make it more difficult for outsiders to see the source code. For security purposes, this implies that outsiders will, at least initially, not see the source code. For competitive purposes, as discussed below in Case Four, this implies that outsiders are limited in their ability, at least initially, to see the source code and use it for derivative or competing works. In these technical and legal ways, proprietary software is at the opposite pole from the open source software discussed in the previous sections.

b) Proprietary software and a monopoly model for information about vulnerabilities

In order to understand the incentives to disclose for security purposes, this chapter begins with a simple monopoly model. When proprietary software is released, one can consider the company to have a monopoly of information about the source code. Employees of the company may not be the *only* persons to have experience with the software at the time of release. For instance, outside alpha and beta testers may have been used for the new software, and the new release may overlap with previous software releases.

---

80 The object code is accessible to authorized users of the software, but the source code is not. For one explanation of the meaning of object and source code, and the relative ease for humans of reading the latter, see *Universal City Studios, Inc. v. Corley*, 273 F.3d 429 at 438–39 (2d Cir. 2001).

Nonetheless, it is helpful to imagine at the moment of release that only employees of the software company have knowledge about the software, including its vulnerabilities.

Consider, next, the incentives that face the company when it learns about a vulnerability. Should the company disclose the vulnerability? Major disadvantages of disclosure would include an immediate reduction in sales of the program and an overall harm to the company's reputation for quality software. Advantages of disclosure would be more indirect — buyers might learn that they could trust the company to disclose vulnerabilities, thus increasing the company's long-term reputation for quality. Another advantage would be similar to the open source model — outside programmers might suggest how to fix the problem. Looking at these disadvantages and advantages of disclosure, a rational corporation might often conclude that it is better to keep the vulnerability secret. This reluctance to disclose vulnerabilities was indeed widespread not long ago in the proprietary software industry.

This simple model at first blush seems quite similar to the breach notification issues discussed in Part C above. The analysis there showed important reasons to support the breach-notification statutes that are now spreading in the US.[81] In both situations, there is a first party (the data holder or software company) and a second-party attacker. In both situations, there are potential harms to third parties (the individuals whose data is lost or the users of the software). For data breaches, the conclusion was that breach-notification statutes are likely appropriate because it is so difficult for individuals to trace harm back to the source of the breach. There are important differences, however, for proprietary software. One difference is that users of software will often (not always) be able to figure out what software program to blame if something goes wrong.[82] A second important difference is that third parties may have important assistance for software vulnerabilities that they did not have in the data-breach setting.

The Security Disclosure Model from "Security Model" helps to show how the monopoly problem tends to erode over time. Even if the software company has a monopoly of information about its product at the time of release, outsiders can and do learn about the software. For standardized

---

81 See Section C(2), above in this chapter.

82 Many computer users have had the experience of a software vendor blaming a problem on the hardware vendor or another software vendor. Even in such cases, the user generally knows that there is a problem and has a short list of suspects for who caused it. Both of these factors are often missing in the breach-of-data scenario.

software products, outsiders can probe and attack the software repeatedly (high *N* for number of attacks). When the software has a vulnerability or otherwise fails to perform appropriately, the outsiders learn about the flaws (high *L*). They can then communicate (high *C*) with other outsiders about the flaws.

To the extent that high *N*, *L*, and *C* thus exist, the monopoly of information may evolve quite rapidly to a competitive situation. Under fully competitive conditions, outsiders would have the same ability to discover and report on vulnerabilities as the software company itself. "Security Model," indeed, used mass-market software as an example of a situation where "security through obscurity" is unlikely to succeed.[83] For the new release of a video game, websites arise almost instantly advising how to "beat the game."[84] This is also the case for new proprietary software programs. Users can and do learn and communicate rapidly about programs after they are released.

To the extent that this analysis is convincing, the rational proprietary vendor is much more likely than it initially appeared to try to discover and fix vulnerabilities. When outsiders predictably discover and publicize flaws, the previous advantages of keeping a secret plummet. Keeping the vulnerability secret not only will likely soon fail, but the company will then have the additional reputation problem of explaining why it did not tell its customers that they were at risk. In such circumstances, it will often be rational for the company to disclose vulnerabilities.

c) The importance of market structure

The discussion thus far has shown some interestingly strong reasons why the market is likely to work well to encourage disclosure of vulnerabilities, even for proprietary software. The next step is to notice that the analysis depends on an assumption that outsiders are effective at discovering and publicizing vulnerabilities. The validity of that assumption will depend on the empirical realities for any given software, and especially on the market structure. One risk is that the closed source in the software will be a strong barrier against outsiders learning about vulnerabilities. A related risk is that skilled and malicious hackers may learn about a vulnerability, but not reveal their knowledge until they use it in a major attack.

The nature of software buyers is important in determining whether a software company will rationally keep a vulnerability hidden. The exist-

---

83 See "Security Model," above note 1 at 181–82.

84 *Ibid.* at 182.

ence of large buyers who care about security increases the likelihood that disclosure will occur. At the extreme, imagine that a software vendor sells to a single buyer (a "monopsonist"), and that the buyer has a high taste for security. What will happen if the software has a serious vulnerability, known to the seller, and the buyer does not know about it? Quite simply, the vendor may lose all its sales.

To choose a vivid example, which, as far as I know, is not true, imagine a company that sells communications software to the US Air Force. Suppose that hackers learn about a vulnerability, and the software company also knows about it but does not disclose its knowledge to the Air Force. Then, on an important mission, the software tells the pilot to turn left when he is supposed to turn right. The mission therefore fails.[85] The Air Force, shall we say, is likely to go ballistic.

This Air Force example brings home the importance to a vendor of pleasing large buyers who care about security.[86] The vendor might then respond with a halfway disclosure, seeking to disclose to its "important" customers but not to its "unimportant" customers. But secrets once shared among many "important" customers are unlikely to stay secret for long.[87] Major proprietary software companies today now disclose vulnerabilities widely, to their own customers and through mechanisms such as the Computer Emergency Response Team (CERT).[88]

The Air Force example highlights the possibility that important customers will get contractual or other legal assurances that they will be notified about vulnerabilities. Although a purchaser of a single desktop does not bargain for such assurances, large customers can and do. The legal assurances might be in an individually negotiated licence to use the software throughout a large organization. For military and other government con-

---

85 If the plane therefore flies into a mountain, that would be a "system crash" of a different sort than usual.

86 In the language of Oliver Williamson, the buyer and seller would have a high level of transaction-specific capital in the use of the software by the organization. Oliver E. Williamson, *The Economic Institutions of Capitalism* (New York: Free Press, 1985) at 30 (discussing transaction-specific capital). The organization would therefore have important incentives to bargain to protect its investment in the software, such as insisting on requirements of disclosure of vulnerabilities.

87 For a discussion of the weaknesses of selective disclosure, see "Security Model," above note 1 at 203.

88 See CERT Coordination Center, online: www.cert.org (providing information about the CERT Coordination Center as a center of Internet security expertise).

tracts, the legal assurance might also come from a statute or other legal mandate from the government.

In addition to buyers with market power and government buyers, another check on under-disclosure is the existence of ethical bug hunters.[89] These individuals or firms seek to find vulnerabilities (bugs) in software. Upon finding a bug, in many instances, they notify the authors of the code in order to provide an opportunity to issue a patch before the vulnerability is publicly announced. To the extent that such bug hunters systematically discover flaws and provide an opportunity to fix them, disclosure will tend to occur even for proprietary software.

d) Historical experience and implications

In considering the likelihood of disclosure for security flaws in mass-produced software, it is helpful to remember the relative youth of the market. The IBM PC was introduced in the early 1980s. Attacks from a distance, including contagious phenomena such as viruses, only became significant with the growth of the Internet in the late 1990s. As late as 1998, the topic of cyber security was practically invisible to government agencies and law professors who were immersed in the law of cyberspace.[90] During the bubble of the late 1990s, many software companies emphasized adding new features and growing quickly instead of writing secure software.[91]

In the past several years, security has generally become a more prominent issue in computing and in mass-market proprietary software in particular. Purchasers, led by large corporations and government agencies, have pushed for and received greater access to the source code from Microsoft and other vendors.[92] An important literature exists within law and economics that shows how effective market discipline by a subset of sophisticated buyers (such as large purchasers of software) can provide a high

---

89 Jennifer Stisa Granick, "The Price of Restricting Vulnerability Publications" (2005) 9 Int'l. J. Comm. L. & Pol'y 1.

90 See Peter P. Swire, "Elephants and Mice Revisited: Law and Choice of Law on the Internet" (2005) 153 U. Pa. L. Rev. 1975 at 1977 note 4 (describing the low level of awareness of Internet security as of the late 1990s).

91 One sign of the shift in industry attitude came in a widely publicized email from Bill Gates to all Microsoft employees on 15 January 2002: "So now, when we face a choice between adding features and resolving security issues, we need to choose security." Email from Bill Gates, Microsoft Chairman & Chief Software Architect, to Microsoft Employees (15 January 2002), online: www.wired.com/news/business/0,1367,49826,00.html.

92 Interview with Dave Ladd, Microsoft Research (28 July 2005).

level of quality for many other buyers (such as mass-market purchasers of software).[93] The strong taste for security from prominent buyers, and the growth of CERT and other disclosure institutions over time thus suggests that the level of disclosure is and will be substantially better for proprietary software in the future than it was in the 1990s.

In conclusion, on the security motive for disclosure for proprietary software, the analysis here does not describe some idealized world where proprietary software vendors always disclose vulnerabilities promptly. Instead, the analysis identifies conditions under which the disclosure is likely to be done relatively effectively: high *N*, *L*, and *C* so that flaws are discovered; buyers with monopsony power who can force disclosure; governments that require disclosure; and effective bug hunters who make the disclosure market more efficient. Where those conditions do not exist, there remains a significant risk that the incentives for vendors to hide their flaws will result in harm to third-party users.

4) Case Four: Competitive Incentives and Proprietary Software

The incentive is clear for why proprietary software companies would keep source code secret. The technological protection of keeping source code hidden complements the legal protection from copyright or other intellectual property regimes. These technological and legal measures raise the costs for competitors to create equivalent software. The proprietary company expects to profit from greater sales of the software.

This incentive to keep source codes hidden is indeed powerful. There are major counter-incentives, however, that often lead proprietary software companies to disclose significant amounts of source code for competitive reasons.

a) Network effects and disclosure for standards

Software companies do not face a simple choice between keeping all source code hidden and revealing all of their code. Instead, there is a continuum. At the closed end of the spectrum, a company can rely entirely on its in-house programmers and not permit anyone else to see any source code. At the opened end of the spectrum, the company can adopt a strong open

---

93 This approach is developed in Alan Schwartz & Louis L. Wilde, "Intervening in Markets on the Basis of Imperfect Information: A Legal and Economic Analysis" (1979) 127 U. Pa. L. Rev. 630 at 662–66.

source licence, and perhaps even make disclosure about consulting and other activities that are outside the scope of that licence.

Choosing where to be on that spectrum requires a complex calculus that has strong elements of game theory. How will other actors react to various levels of openness? Being more open has the potential advantage of attracting other developers. If enough developers work with the software program, then a critical mass might result. There can thus be "network effects," where "the value that consumers place on a good increases as others use the good."[94] However, being more open has the potential disadvantage that the other participants may get sales from participation with your software. At the extreme, this sounds like the complaint quoted earlier from the open source programmer who gets, at most, "a few crumbs" from the code she has written.[95] In essence, the software company is trading-off the expanded size of the market that can come from openness versus the diminished share of that market that can come from the participation of the other competitors.

A recent announcement by VMware shows this dynamic at work. In August 2005, *The New York Times* reported: "VMware, the leader in the fast-growing market for virtual machine software, plans to announce today that it will share its code with partners like I.B.M., Intel and Hewlett-Packard in an effort to make the VMware technology an industry standard."[96] The article stated, "The move by VMware is an attempt to defend against Microsoft before that company accelerates its drive into virtual machine software."[97]

The VMware announcement nicely illustrates four points already made in this chapter. First, disclosure is on a continuum between an open and a closed source: "The partners will be able to modify the basic code for their own products, to be sold to customers. But the VMware program does not go so far as to allow the code to be freely distributed to anyone, as is the case with Open Source software projects."[98] Second, openness is a strategy to expand the overall market for a software project. In the words of

---

94 Mark A. Lemley & David McGowan, "Legal Implications of Network Economic Effects" (1998) 86 Cal. L. Rev. 479 at 481. Lemley & McGowan provide a comprehensive examination of the legal implications of such network effects, including a number of software-related issues.

95 See Posting on ZDNET, above note 72 and accompanying text.

96 Steve Lohr, "VMware to Share Its Code: Hoping to Be the Standard" *New York Times* (8 August 2005) C6 [Lohr].

97 *Ibid.*

98 *Ibid.*

an IBM executive: "This is a move toward open standards, and that is the path toward accelerating market growth and innovation."[99] Third, the shift to openness is accompanied by a strategy for the proprietary company to continue to reap financial rewards. VMware's chief executive Diane Greene said she "sees continued growth opportunities by adding features and services on top of its basic technology."[100] This quote matches the discussion above of open source competitive strategies: stay ahead of the curve; provide services where competitors do not learn easily about innovations; and, in general, seek a significant percentage of the growing market that uses VMware code. Fourth, complex game-theory calculations determine the optimal degree of openness for the software company. VMware is trying to stay closed enough to reap financial awards, but it is being pushed by other large players to become more open: "VMware's big industry partners have become increasingly concerned that it is becoming too powerful, and could potentially become a crucial, proprietary layer of software in data centers, much as Microsoft's Windows rules the market for personal computer operating systems."[101] In short, if VMware had not decided to become more open, the big industry partners may have abandoned it and gone elsewhere.

One way that this dynamic often plays out is through the standards process. The "effort to make the VMware technology an industry standard"[102] highlights the way that companies often face the question of how much to disclose; what should go into an open standard versus what should be kept in-house? My discussions with market participants reveal a high level of awareness about the importance of this standards issue. I have not found, however, any persuasive general account of how to decide how much to disclose in the standards process.[103]

For purposes of this chapter, I suggest two thoughts about the standards issue. First, because of the importance of the interactions among a few or several players, the optimal analysis is likely to depend on game theory.[104] Second, the rational level of disclosure for a vendor will often

---

99 *Ibid.* (quoting Susan Whitney, a general manager in IBM's server business).

100 *Ibid.*

101 *Ibid.*

102 *Ibid.*

103 For a thorough examination of legal issues arising from standard-setting organizations, see generally Mark A. Lemley, "Intellectual Property Rights and Standard-Setting Organizations" (2002) 90 Cal. L. Rev. 1889.

104 See Martin Shubik, *Game Theory in the Social Sciences: Concepts and Solutions* (Cambridge: MIT Press, 1982) at 300. Shubik explained that "[f]ew-player games are ones in which the players number anywhere from three to around twenty." *Ibid.*

be at an intermediate position between the extremes of open and closed source. This intermediate position is consistent with a theme of this chapter: movement along the continuum between an open and a closed source is a strategically and theoretically important topic.

b) Developer mindshare and "get them while they're young"

Proprietary software companies have an incentive to keep the source code in-house and only among software writers who have signed non-disclosure agreements. In this way, the company keeps maximum control over who gets to work with the code and can manage how all aspects of the code are developed over time. The number of software writers is then limited to the number of persons hired by the company.

Companies simultaneously face an incentive to get the maximum number of skilled software developers to work with the code. Greater developer "mindshare" leads to greater innovation on the company's code. Developer enthusiasm for this software, in turn, may reduce the amount of developer innovation that goes into competing software offerings.

Taking the conflicting incentives together, we see that a proprietary software company would rationally weigh the profits that come from increased control against the profits that result from innovation and other activity by those outside of the firm.

Two strategies illustrate how companies seek to resolve the conflicting incentives. First, companies can try to "get them while they're young." Apple Computer has famously placed its computers in elementary and other schools, hoping to train a new generation of loyal users.[105] Similarly, Microsoft has been licensing significant and increasing amounts of source code to universities since the early 1990s.[106] Computer science professors and their graduate students thus get to work directly on the source code. If programmers learn the intricacies of a software program during their student days, then it seems more likely that they will continue to work with that code after they leave school. Second, companies can release code to de-

---

He added, "Oligopoly theory, the formation of political coalitions, international negotiations, ecological struggles, and cartel problems all fall into this category." *Ibid.* His examples of oligopoly theory and cartel problems are closely related to the industry-structure issues implicated by the decision of companies about whether to cooperate (share source code) or compete (keep source code hidden).

105 Apple.com, "25 Years of Education Experience," online: www.apple.com/ca/education/whyapple/25years.html (charting Apple's investments in education from 1980 to the present).

106 Interview with Dave Ladd, above note 92.

velopers in the hopes of having the outside developers engage more deeply with the software. This strategy was used by Microsoft when it launched the Shared Source Initiative, under which it has released considerably more source code.[107]

c) Overcoming collective action problems

Proprietary software companies may use open approaches for another reason: to overcome collective action problems. Mancur Olson, a pioneer in researching such issues, memorably explained that for groups with common interests "there is a systematic tendency for 'exploitation' of the great by the small!"[108] Olson analyzed the incentives for large and small actors who would share in the benefit of a collective good.[109] His key point is that the large actor may have a rational basis for investing in the collective good because its own return from the investment may be large enough to justify the cost. A small actor, by contrast, would not get a large enough return from its own investment to justify the action. In such cases, the small actors would rationally be free riders, not investing even in a good that benefits all concerned. Thus, only the large actors will invest in the public good — the "exploitation" that Olson described.[110]

Turning to software, proprietary firms in some circumstances may find it rational to invest in open software in order to solve a collective-action problem. A notable current example would be an improved system for authentication on the Internet.[111] Put simply, it would be a great help to e-commerce and many other activities on the Internet for each of the two parties

---

107 For a discussion of the Shared Source Initiative, see Microsoft Shared Source Initiative, online: www.microsoft.com/resources/sharedsource/Initiative/Initiative.mspx.

108 Mancur Olson, *The Logic of Collective Action; Public Goods and the Theory of Groups* (Cambridge: Harvard University Press, 1965) at 29 [emphasis omitted] [footnote omitted].

109 Olson defines a public good or collective good "in terms of [the] infeasibility of excluding potential consumers of the good." *Ibid.* at 14 note 21. Modern definitions stress that a public good is not only non-excludable but also non-rival (i.e., use by one person does not reduce use by another). Robert P. Merges *et al.*, *Intellectual Property in the New Technological Age*, 3d ed. (New York: Aspen Law and Business, 2003) at 11–12. A classic example of a public good is clean air. If a factory puts filters on its smokestacks, reducing pollution, it has no feasible way to exclude potential consumers of the good, those who breathe it. *Ibid.*

110 Olson, *ibid*. at 28–29.

111 Effective privacy protections do not necessarily preclude improved authentication. For an excellent National Academies of Science report on the topic, see National Research Council Committee on Authentication Technologies and their Privacy Im-

to be confident of the identity of the other party. Sellers and credit card companies have long faced the risk that buyers were using false identities or credit cards. More recently, the problem has grown more acute for buyers, who can no longer trust that the apparent seller's site is what it seems to be. Ordinary individuals today face the problem of phishing, defined as "the process of tricking or socially engineering an organisation[']s customers into imparting their confidential information for nefarious use."[112] A growing threat is pharming, defined as manipulation of the way that a customer "locates and connects to an organisation's named hosts or services through modification of the name lookup process."[113]

The lack of good authentication poses risks not only to websites and Internet users, but also to the overall growth of computing and the Internet. Large companies who benefit from this overall growth thus may have an incentive to take actions that will improve authentication.

The actions of Microsoft in the Internet-authentication area illustrate the way that this collective-action problem might be addressed either by a proprietary or open approach. Microsoft's first big authentication effort was heavily proprietary. Its Passport program, as proposed in 2000, contemplated that users would have a unique username that Microsoft would issue.[114] To make the system even more proprietary, data about sites the user visited would go through the Microsoft system.

The reaction to Passport was negative and overwhelming—on security, privacy, and competitive grounds.[115] A main security problem was that

---

plications, *Who Goes There? Authentication through the Lens of Privacy* ed. by Stephen T. Kent & Lynette I. Millett (Washington, DC: National Academies Press, 2003).

112 Gunter Ollman, *Next Generation Security Software Ltd., The Phishing Guide: Understanding and Preventing Phishing Attacks* (2004) at 3, online: www.ngsconsulting.com/papers/NISR-WP-Phishing.pdf. I am currently serving as reporter for a process convened by the National Consumers League on strategies for addressing phishing.

113 Gunter Ollman, *Next Generation Security Software Ltd., The Pharming Guide: Understanding and Preventing DNS-Related Attacks by Phishers* (2005) at 3, online: www.ngsconsulting.com/papers/ThePharmingGuide.pdf.

114 See Electronic Privacy Information Center, online: www.epic.org/privacy/consumer/microsoft. Passport was initially proposed as part of broader Microsoft initiatives called "Hailstorm" and ".NET." See *ibid.*

115 See, for example, SOA World Magazine News Desk, "WSJ Exclusive Interview: Single Sign-On is a Single Point of Failure, Says EPIC Counsel" (1 January 2000), online: http://webservices.sys-con.com/read/39389.htm [SOA World Magazine News Desk] (discussing the privacy risk of central authentication). The Federal Trade Commission also initiated a complaint on security grounds, stating that Microsoft's disclosures about security were deceptive under section 5 of the *Federal Trade Com-*

Microsoft's role at the centre of the system created a single possible point of failure — if the Microsoft system was hacked, then all of the identifying information could be compromised. On privacy, advocacy groups sharply criticized the amount of identifiable data that would be gathered,[116] and the EU brought a major privacy complaint.[117] On competitive grounds, online retailers and other key players did not want Microsoft to have the "crown jewels" of electronic retailing (e-tailing), namely information about identified persons, including where they surfed and shopped.[118] The original Passport plans were halted, and a scaled-down Passport system has been used principally to sign into Hotmail, MSN, and other Microsoft-specific activities.[119]

In late 2005, Microsoft announced an entirely different approach to Internet authentication based on openness.[120] The new InfoCard approach

---

mission Act, 15 U.S.C. § 34 (2000). See Federal Trade Commission, Press Release, "Microsoft Settles FTC Charges Alleging False Security and Privacy Promises" (8 August 2002), online: www.ftc.gov/opa/2002/08/microsoft.htm (discussing Microsoft's agreement to settle the FTC's "charges regarding the privacy and security of personal information collected from consumers" through Passport); see also Complaint para. 21, *In re Microsoft Corp.*, No. 012-3240 (8 August 2002), online: www.ftc.gov/os/caselist/0123240/microsoftcmp.pdf ("The acts and practices of respondent as alleged in this complaint constituted unfair or deceptive acts or practices in or affecting commerce in violation of Section 5(a) of the *Federal Trade Commission Act*.").

116 See, for example, SOA World Magazine News Desk, above note 115 ("'It's bad practice to create personally identifiable records unless it's necessary. Microsoft, through Passport, is creating personally identifiable records.'" (interviewing Chris Hoofnagle, legislative counsel for Electronic Privacy Information Center (EPIC)).

117 See Paul Meller, "Microsoft to Alter Online System to Satisfy Europe" *The New York Times* (31 January 2003), online: http://query.nytimes.com/gst/fullpage.html?res=9503EFDE1638F932A05752C0A9659C8B63 (reporting that Microsoft agreed to make "radical" changes to .Net Passport to "avert a clash with European regulators over data privacy").

118 Byron Acohido, "Microsoft, Banks Battle to Control Your E-Info" *USA TODAY* (13 August 2001), online: www.usatoday.com/tech/news/2001-08-13-microsoft-banks-e-info.htm.

119 For the current privacy policies for Passport, see "Microsoft.com Privacy Statement," online: www.microsoft.com/info/privacy.mspx#E6D.

120 For a good overview of this approach, called InfoCard, see Robert McMillan, "InfoCard Not Son of Passport, Says Microsoft Executive" *INFOWORLD* (21 September 2005), online: www.infoworld.com/article/05/09/21/HNinfocard_1.html [McMillan], who notes that, unlike Passport, "InfoCard is being designed to work on client and server software that was not developed by Microsoft." As in "Security Model," I note that I am a member of Microsoft's Trustworthy Computing Academic Advisory

is designed to make it easy for users to keep track of their passwords and other credentials,[121] while also making it far harder to impersonate a buyer or seller. InfoCard is a standard (or a set of standards) rather than a solution provided by one company. The approach is designed to be open on the client side — working with browsers such as Firefox and Opera — and on hardware running Apache, Linux, and other systems.[122] Unlike the original Passport, Microsoft would not see the data that moves between surfers and websites. In addition, Microsoft has announced it will not charge for its work on InfoCard.[123]

The suggestion I make here is that the InfoCard project is an illustration of what Mancur Olson described as the "'exploitation' of the great by the small."[124] It would benefit a very wide range of players to have effective authentication on the Internet, so that a wide variety of surfers and websites can interact with less risk of spoofing and fraud. It will likely take large investment by one or more major players, however, to make a dent in the collective-action problem of how to authenticate such a range of surfers and websites. Microsoft showed with Passport that it would prefer to have a proprietary approach to solving the authentication problem. That effort did not succeed. As a next-best solution, Microsoft is now investing resources in the open InfoCard process.

It is useful to compare the ways in which InfoCard is similar to or different from the VMware account discussed above. The similarity is based on the role of standards and the way that critical mass may develop where there is openness in development. The difference essentially boils down to the extent that there is a private good versus a public good. For VMware, the goal was to create a private good so that its software would succeed

---

Committee, which is a group of nineteen academics that has been asked to provide advice on security and privacy issues to Microsoft "Security Model," above note 1 at 198 note 51. I have heard presentations on InfoCard both in public, and in connection with that committee's work. The views expressed in this chapter are entirely my own, and I have not been compensated by Microsoft in connection with this research. I have also spoken about the issues in this chapter at great length with many open source supporters.

121 McMillan, *ibid*.

122 *Ibid.*

123 See Mary Branscombe, "Credit Where It's Due" *The Guardian* (9 June 2005), online: www.guardian.co.uk/microsoft/Story/0,,1501893,00.html.

124 Olson, above note 108 at 29 [emphasis omitted]. A similar argument would support the investment in another authentication project, the Liberty Alliance, by a coalition of large companies engaged in e-commerce, online: www.projectliberty.org.

competitively against Microsoft. VMware strategically chose the degree of openness in order to gain allies such as IBM and HP, and to get direct profits based on "adding features and services on top of its basic technology."[125] By contrast, InfoCard seems designed to create a public good, which is non-excludable (anyone can use the standard) and non-rival (use by one person does not limit use by another). Speakers for Microsoft have stressed instead the goal of growing the entire space of e-commerce and computing, rather than forcing users to buy Microsoft for authentication.[126] The incentive for Microsoft is that it will gain enough new sales from overall growth to justify its investment in creating InfoCard.[127]

d) Convergence of open source and proprietary approaches

In summary of Case Four, there are complicated interactions that affect when a proprietary software company will reveal source code for competitive reasons. Significant incentives exist for such disclosure, including: the possibility of network effects; the strategic use of open standards; the enticement of students and other developers; and the possibility of overcoming collective-action problems. Taken together, there are major incentives for proprietary software to shift toward disclosure, just as there are major incentives for open source software to use secrecy in substantial ways. The overall trend appears to be toward substantial convergence of the two previously separate approaches.

5) Case Five: Security Incentives and Government Agencies

From the discussion of open source and proprietary software, we now shift to examination of the incentives for government agencies to disclose or not. Case Five examines the incentives to disclose in connection with promoting security. Case Six looks at the competitive incentives of government agencies and focuses on agency incentives to protect and expand their own turf.

---

125 Lohr, above note 96.

126 Microsoft Corporation, "Microsoft's Vision for an Identity Metasystem" (May 2005), online: http://msdn.microsoft.com/en-us/library/ms996422.aspx.

127 Note that the Mancur Olson argument shows why the large company may invest substantial funds in the creation of a public good. The investment in the public good may still be less than is societally optimal.

a) Government agencies and national security

To begin, the government agency can be modeled similarly to the proprietary software company. The agency has information about its own systems and activities. The agency has various incentives to keep information secret, including the hope that attackers will not learn about any vulnerabilities. The agency also has various incentives to disclose information, including the gain the agency may realize if more defenders know and respond to the information.

As discussed in "Security Model," there are often good reasons for categories of military and intelligence information to be kept secret. First, the high cost of each military attack, including the risk of casualties, means the number of attacks (*N*) is often low. In such instances, secrecy is often a rational strategy.[128] Second, many government secrets concern sources and methods, and other aspects of surveillance where secrecy is, again, rational.[129] Third, the chain of command and hierarchical nature of the military often make it more feasible to disclose to allies selectively, without disclosing to all parties.[130]

Related to the low *N* is the fact that significant portions of government activities are different from activities in the private sector. For example, nuclear launch codes and stealth aircraft are not (presumably) used in the private sector. To the extent that government activities are different, attackers gain less experience from attacking private-sector systems. In such situations, there is low *N* and *L*, and secrecy is more likely to be rational. Even if an agency knows about a vulnerability, the benefits of non-disclosure to the attackers quite possibly outweigh the benefits of disclosure to the defenders.

b) Information sharing and third-party effects

There are other government activities where the calculus is more complex. Consider the incentives for an agency, such as the FBI, to disclose information about suspected terrorists to state and local agencies and private-sector actors. The various third parties might help the FBI by spotting and arresting the suspects, but disclosure to the third parties might also help the terrorists by tipping them off that their identities are known. In this infor-

---

128 See "Security Model," above note 1 at 177 (commenting that hiddenness will benefit military defenders because the cost of each attack will lower the total number of attacks and prevent the attackers from taking advantage of increased knowledge about the defences).

129 *Ibid.* at 191–93.

130 *Ibid.* at 203–4.

mation-sharing instance, disclosure quite possibly will help both attackers and defenders, and it is difficult to generalize about when to disclose.

In other instances, the third parties may receive clear benefits from disclosure, but the first party may not share in those benefits. Consider the possibility that the FBI learns that a certain type of attack is likely to occur, such as a nighttime attack on a power plant in an American or foreign city. The FBI may decide not to disclose because it places a higher priority on continuing its investigation of the entire terrorist group. The area that is attacked, however, is likely to place a much greater value on damage to its city.[131] In this example, the benefits of disclosure to the city may be external to the FBI's calculus. The FBI may err on the side of less disclosure than would have occurred if the localities and the FBI had made the decision jointly.

This possibility of external, negative effects on third parties can occur much more generally as a result of government action. One sad and dramatic example is the way that environmental decisions were made in the Soviet Union before its dissolution. Essentially, government agencies made decisions based on their incentives to meet production quotas and other government goals. Meanwhile, very serious environmental spillover effects often occurred. Due to one especially large disaster, the entire Aral Sea now appears to be on course to dry up, with numerous secondary consequences.[132]

c) Public choice and the weak constraints on government secrecy

The potentially large effect of government secrecy on third parties is analogous to the potentially large effect of proprietary software secrecy on third-party users. For proprietary software, the discussion above identified a number of mechanisms that push toward disclosure of vulnerabilities, even for companies that initially prefer to keep the vulnerabilities secret. These mechanisms included: (i) high N, L, and C so that flaws were discovered; (ii) buyers with monopsony power who could force disclosure; (iii) governments that required disclosure; and (iv) effective bug hunters who made the disclosure market more efficient.

---

131 The locality, in this example, is analogous to the users of proprietary software who would benefit from the disclosure of vulnerabilities by the software creator. Both the locality and the software users can suffer negative effects due to the incentives facing the first party.

132 For discussion of the environmental effects of Soviet-era decision making on the Aral Sea, see Nicola Jones, "South Aral 'Gone in 15 Years'" *New Scientist* (19 July 2003) at 9; Joshua Calder & Jim Lee, "Inventory of Conflict and Environment, Case Study Number 69: Aral Sea and Defense Issues" (1995), online: www.american.edu/ted/ice/aralsea.htm.

The question is the extent to which those market mechanisms have an analogy for government agencies. Public choice theory is the branch of political economy that analyzes political institutions by use of economic theory.[133] In some instances, agencies and other institutional actors may get to good outcomes because the organizational incentives are well-designed. When it comes to the optimal level of disclosure for vulnerabilities, however, the mechanisms to force disclosure seem much less likely to be successful than for proprietary software. The number of attacks and learning by attackers will often be less due to the backdrop of national security and secrecy. Rarely will there be any equivalents of monopsony purchasers who can force disclosure.[134] Laws such as the *Freedom of Information Act*[135] are indeed a crucial mechanism for creating openness and accountability, but discovery of any specific vulnerability depends at best on the serendipity of whether someone makes the right *Freedom of Information Act* request. Finally, bug hunters are discouraged in various ways in the public sector: stealing classified information is a crime; there are legal and personnel sanctions against those who "leak"; and whistleblowing is typically risky to do.

In short, it is difficult to be optimistic about the mechanisms in place to ensure the optimal level of disclosure from government agencies. As the next section shows, the magnitude of the problem is likely even greater when one considers an agency's competitive incentives.

6) Case Six: Competitive Incentives and Government Agencies

When examining incentives to disclose, Case Five shows that government agencies would consider the effects on their own security goals but would not internalize the effects on third parties. Those risks are magnified when one considers the overall competitive position of a government agency.

---

133 See Daniel A. Farber & Philip P. Frickey, *Law and Public Choice: A Critical Introduction* (Chicago: University of Chicago Press, 1991) at 1 and 6–7 (defining public choice theory).

134 For instance, it seems unlikely that foreign or American localities will be effective at forcing the FBI or other federal agencies to disclose information when the FBI does not believe it is in the Bureau's best interest to do so. One partial exception, where the incentive effects lead to fuller disclosure, is when the Presidency and Congress are held by different parties. My experience in the Clinton Administration was that the Republican Congress vigorously investigated even small alleged problems. The level of oversight fell sharply when the same Republican majority was in office with a Republican, President George W. Bush, in the White House,.

135 5 U.S.C. § 552 (2002).

A prominent strand of public choice theory posits that agencies will seek to maximize "turf."[136] That is, agencies will have goals such as expanding their budget, maximizing their flexibility vis-à-vis other institutions, avoiding embarrassment, and so on. One example is when different law-enforcement agencies compete to get credit on a high-profile case. There have long been laments in law enforcement that information sharing (i.e., disclosure) is lacking because agents do not want to lose control of a case.[137]

Under this model, consider the incentives of a government agency when there is a known problem, such as a vulnerability, in a project where the agency is supposed to be in charge. The disadvantages of disclosure are evident — disclosure will expose problems in the agency's area of responsibility. There could well be hostile public hearings, as well as possible reductions in budget and agency discretion. The advantages of disclosure may be only indirect. If the embarrassing details will come out later, it may make sense to disclose the problem earlier, spun in a way that favours the agency. In many instances, these indirect benefits of disclosure are small. The incentive of government agencies in many situations, when they know about a vulnerability or mistake, is to deny, deny, deny.

To what extent are there market or similar mechanisms to correct for this tendency to err on the side of secrecy? Some such mechanisms can exist. An opposing political party has the incentive to expose corruption or wrongdoing in the current administration. Depending on the degree to which it is beholden to the current administration, the media may have incentives to "break a story" that shows mistakes or wrongdoing in government. The analysis here is consistent with the grand tradition of checks and balances because opposition political parties and a free press set limits on secrecy and wrongdoing by incumbents.

On a more day-to-day level, however, the political opposition or reporters may not be able to learn about many agency secrets. In that instance, it is good policy to have additional accountability institutions. In the US federal government, such institutions include the following: inspector generals in each agency; reports from the General Accountability Office;

---

136 See, for example, William A. Niskanen, Jr., *Bureaucracy and Representative Government* (Chicago: Aldine-Atherton, 1971) at 111–12 (arguing that bureaus seek to provide the broadest possible range of services in order to garner the most power possible).

137 For one article that gives a vivid flavour of ongoing turf wars at the federal and state level, see Mike Kelly, "State Counter-Terror Organization in Disarray" *The* (*Bergen County, NJ*) *Record* (9 October 2005) O1.

congressional hearings; the *Freedom of Information Act*;[138] and other open-government requirements. Creation of these institutions and these laws is crucial. Without them, there will undoubtedly be too much secrecy and self-protection by agencies. Even with them, I believe that agencies err on the side of secrecy and excessive use of classified documents far too often.[139]

In response, I believe there should be ongoing and significant efforts to detect and correct areas of excessive government secrecy. For instance, I have proposed modifying the "gag rules" under the foreign intelligence surveillance laws.[140] More generally, this entire project on security and obscurity has developed systematic approaches to describing and assessing the limited circumstances under which obscurity actually helps security.

E. CONCLUSION

An organizing theme of this chapter has been that the incentives to disclose are based on two distinct calculations. How does disclosure help or hurt security? How does disclosure help or hurt the organization competitively? Discussions about disclosure of vulnerabilities have typically failed to address the competitive issues. Discussions about the economics of open source versus proprietary software have typically failed to address the security disclosure issues. Understanding of the incentives to disclose or hide requires attention to both questions.

For the open source movement, one theme of this chapter has been to identify and highlight ways that secrecy is used for security and competitive reasons, despite the ideology of openness. The repeated use of the term "secret sauce" is suggestive here — even cute. The image of the sauce is a small layer of secrecy on top of the main course, which is not secret. The secret sauce is tasty, adding zest and individuality to each dish. The term suggests that the use of secrecy is, after all, a minor ingredient.

---

138 Above note 135.

139 For one excellent source on overclassification, including quotations about the extent of the problem, see US, *Emerging Threats: Overclassification & Pseudoclassification: Hearing Before the Subcommittee on National Security, Emerging Threats, and International Relations of the House Committee on Government Reform*, 109th Cong. (2005) (statement of Thomas Blanton, Executive Director, National Security Archive, George Washington University) at 121–24, online: www.fas.org/sgp/congress/2005/030205overclass.html.

140 Peter P. Swire, "The System of Foreign Intelligence Surveillance Law" (2004) 72 Geo. Wash. L. Rev. 1306 at 1356–60.

I speculate that use of the term "secret sauce" is a linguistic effort to downplay the use of secrecy, in recognition that any lapse from openness risks being viewed as a shameful act among the open source community. To the extent that the analysis here shows multiple, significant ways that secrecy-for-security and secrecy-for-competitiveness coexist with open source projects, the linguistic effort may be a sign of guilty consciences among true believers who nonetheless employ secrecy. The ideology of openness is not necessarily a good match for how the rational open source programmer acts.

Do I write this out of some admiration or preference for secrecy over openness? Not at all. I am a great supporter, for instance, of the *Freedom of Information Act*, openness in government, and robust rights of a free press. For breach notification, I support measures to ensure that third parties are not harmed by the externality described in this chapter. For software vulnerabilities, I applaud the many measures taken in recent years that have accelerated discovery, notice, and patch-creation for vulnerabilities. Where there is evidence that the incentives to disclose and fix vulnerabilities are not working, further measures should be explored.

This chapter, instead, has been primarily descriptive. It began with an attempt to explain when it is correct to say "there is no security through obscurity" and when instead it makes sense to say that "loose lips sink ships." "Security Model" examined when disclosure would help or hurt security, considering the effects on all the relevant actors. This work took on the next task, identifying the incentives for key actors to disclose or not, for both security and competitiveness reasons. Together, the hope is to create systematic ways to decide when it makes sense to disclose or keep a secret. In a world of pervasive new information flows and systems, that is no small thing.