A Model for When Disclosure Helps Security: What Is Different About Computer and Network Security?

Peter Swire

Forthcoming, Journal on Telecommunications and High Technology Law (vol. 2, 2004)

Public Law and Legal Theory
Working Paper Series
No. 17

Center for Law, Policy and Social Science
Working Paper Series
No. 12

July 17, 2004

"A Model for When Disclosure Helps Security: What Is Different About Computer and Network Security?"

Peter P. Swire*

TABLE OF CONTENTS

* Professor of Law and John Glenn Research Scholar in Public Policy Research, Moritz College of Law of the Ohio State University. Many people from diverse disciplines have contributed to my thinking on the topic of this paper. An earlier version of the project, entitled "What Should Be Hidden and Open in Computer Security: Lessons from Deception, the Art of War, Law, and Economic Theory," was presented in 2001 at the Telecommunications Policy Research Conference, the Brookings Institution, and the George Washington University Law School, *available at* www.peterswire.net. I am also grateful for comments from participants more recently when this topic was presented at the Stanford Law School, the University of North Carolina Law School, and the Silicon Flatirons Conference at the University of Colorado School of Law. Earlier phases of the research were supported by the Moritz College of Law and the George Washington University Law School. Current research is funded by an award from the John Glenn Institute for Public Service & Public Policy.

Introduction

This Article asks the question: "When does disclosure actually help security?" The question of optimal openness has become newly important as the Internet and related technologies have made it seem inevitable that information will leak out. Sun Microsystems CEO Scott McNealy received considerable press attention a few years ago when he said: "You have zero privacy. Get over it."[1] An equivalent statement for security would be to say: "You have zero secrecy. Get over it." Although there is a germ of truth in both statements, neither privacy nor secrecy is or should be dead. Instead, this Article seeks to provide a more thorough theoretical basis for assessing how disclosure of information will affect security. In particular, this Article seeks to understand what is different between traditional security practices in the physical world, on the one hand, and best practices for computer and network security, on the other.

The discussion begins with a paradox. Most experts in computer and network security are familiar with the slogan that "there is no security through obscurity."[2] For

[1] A. Michael Froomkin, "The Death of Privacy," 52 Stanf. L. Rev. 1461, 1462 (2000).
[2] A search on Google for "security obscurity" discovered 101,000 web sites with those terms. Reading through the web sites show that a great many of them discuss some version of "there is no security through obscurity."

proponents of Open Source software,[3] 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 II slogan says "loose lips sink ships."[4] Most experts in the military and intelligence areas believe that secrecy is a critical tool for maintaining security.

Part I of the Article provides a basic model for deciding when the Open Source and military/intelligence viewpoints are likely to be correct. Insights come from a 2x2 matrix. 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. The second variable is the extent to which the disclosure is likely to improve the defense. Disclosure might help the defense, notably, by teaching defenders how to fix a vulnerability and by alerting more defenders to the problem. The 2x2 matrix shows the interplay of the help-the-attacker effect and help-the-defender effect, identifying four basic paradigms for the effects of disclosure on security: the Open Source paradigm; the Military/Intelligence paradigm; the Information Sharing paradigm; and the Public Domain.

Part II provides an explanation of why many computer and network security issues are different from military and other traditional security problems of the physical world. The discussion focuses on the nature of the "first-time attack" or the degree of what the paper calls "uniqueness" in the defense. Many defensive tricks, including secrecy, are more effective the first time there is an attack on a physical base or computer system. Secrecy is far less effective, however, if the attackers can probe the defenses repeatedly and learn from those probes. It turns out that many of the key areas of computer security involve circumstances where there can be repeated, low-cost attacks. For instance, firewalls, mass-market software, and encryption algorithms all can be

---

[3] Wikipedia, an on-line encyclopedia that uses Open Source approaches, defines "open source" as "a work methodology that fits the Open Source Definition, and generally is any computer software whose source code is either in the public domain or, more commonly, is copyrighted by one or more persons/entities and distributed under an open-source license such as the GNU General Public License (GPL). Such a license may require that the source code be distributed along with the software, and that the source code be freely modifiable, with at most minor restrictions." http://en.wikipedia.org/wiki/Open_source (last visited July 16, 2004).

Wikipedia states: "**Source code** (commonly just **source** or **code**) refers to any series of statements written in some human-readable computer programming language. In modern programming languages, the source code which constitutes a software program is usually in several text files, but the same source code may be printed in a book or recorded on tape (usually without a filesystem). The term is typically used in the context of a particular piece of computer software. A computer program's *source code* is the collection of files that can be converted from human-readable form to an equivalent computer-executable form. The source code is either converted into executable by an software development tool for a particular computer architecture, or executed from the human readable form with the aid of an interpreter." http://en.wikipedia.org/wiki/Source_code (last visited July 16, 2004).

[4] For images of World War II posters on the subject, see http://www.state.nh.us/ww2/loose.html. The posters tell vivid stories. One poster has a picture of a woman and the words "Wanted for Murder: Her Careless Talk Costs Lives". Another shows a sailor carrying his kit, with the words "If You Tell Where He's Going … He May Never Get There".

attacked repeatedly by hackers. Under such circumstances, a strategy of secrecy – of "security through obscurity" – is less likely to be effective than for the military case.

Even recognizing the lower effectiveness of secrecy in many computer and network applications, there will still often be advantages of secrecy in practice. Part III relaxes the assumptions of the model presented in Part I. The Open Source approach makes three assumptions: (1) disclosure will offer little or no help to attackers; (2) disclosure will tend to upgrade the design of defenses; and (3) disclosure will spread effective defenses to third parties. In practice, secrecy will often be of greater use than the Open Source advocates have stated, because one or more of the three assumptions will not hold. Part III explains some of the major categories of situations where secrecy is likely to be more or less effective at promoting security.

The chief intellectual task of this Article is to help us think about when disclosure will help or harm security. There are other major considerations that go into an informed judgment about whether to disclose information about a security vulnerability. For instance, it may promote accountability and the long-run health of the system to err on the side of disclosure. This instinct underlies the Freedom of Information Act[5] and many other laws and practices encouraging disclosure. As another example, disclosure can compromise personal privacy in some circumstances. Accountability and privacy are vital goals in the overall analysis of when to disclose information. Discussion of those goals figures prominently in my larger research project on openness and security. In this Article, however, the focus is on when disclosure will help the specific goal of system security: when will disclosure protect against the attacker gaining control of a physical installation or computer system.

I. A Model for When Disclosure Helps Security

When does disclosure help security? The intuition for experts in the military and intelligence realms is usually that secrecy (the lack of disclosure) is an essential tool for enhancing security. Military bases and weapon systems are cloaked in secrecy. Intelligence agencies tell little about their capabilities, sources, and methods. The slogan for this position is the World War II motto that "loose lips sink ships." The graphic image is that too much disclosure ('loose lips") will tip off the enemy where to send its submarines ("sink ships").[6] In such instances, disclosure can be tantamount to treason.

[5] *See* 5 U.S.C. § 552, as amended by Pub. L. No. 104-231, 110 Stat. 3048 (1996).
[6] *See supra* note 3.

A. Case A: The Open Source Paradigm.

Despite the World War II intuition, a pervasive theme of many computer security discussions is that "there is no security through obscurity."[7] For people outside of the computer security realm, it may initially be difficultto understand how that slogan has become a truism. Based on research and discussions with computer security researchers, there seem to be three assumptions, often implicit, that undergird the slogan. In considering whether to disclose a vulnerability, supporters of openness seem to assume the following:

(A1) Attackers will learn little or nothing from the disclosure.
(A2) Disclosure will prompt the designers to improve the design of defenses.
(A3) Disclosure will prompt other defenders to take action.

The discussion below in Parts II and III will develop in more detail the intuitions that underlie these three assumptions. It will also critically examine each assumption. For the present, however, the basic idea for assumption (A1) is that software and network vulnerabilities, once discovered by any attacker, will often quickly become known to other attackers. For instance "warez" sites and other mechanisms exist to teach hackers about new attacks.[8] Public disclosure of a vulnerability will thus not significantly help attackers exploit the vulnerability.

The basic idea for assumption (A2) is a deeply-held tenet in the Open Source movement. The idea is that software will improve quickly if a wide array of programmers can see the code, find flaws in it, and fix those flaws. In the words of researchers Randy Bush and Steven Bellovin: "Hiding security vulnerabilities in algorithms, software, and/or hardware decreases the likelihood they will be repaired."[9]

The basic idea for assumption (A3) is that many people may be affected by a vulnerability other than the software or system designers. For a software program, for instance, assumption (A2) is directed at the group of programmers who may write new code to improve the software. There are likely many system owners, however, who use the software program but are not involved in writing it. Assumption (A3) focuses on how disclosure of a vulnerability can improve the security of these system owners. System owners who learn of the vulnerability can install a patch or upgrade once it is available.

---

[7] *See supra* note 2. The origin of the slogan "there is no security through obscurity" is obscure. I would welcome information on who coined the term. It was certainly used by the early 1990's. *See, e.g.*, *Netware Users React to Security Threat*, Internet Week, 2 (Oct. 5, 1992) (Rop Gonggrijp refers to "security through obscurity" as a policy used by Novell).

[8] *E.g.*, http://easywarez.com; http://ICEWAREZ.net (examples of "warez" sites that provide downloads of software illegally, including software that can be used for hacking purposes).

[9] *E.g.*, Randy Bush & Steven M. Bellovin, "Security Through Obscurity Dangerous," Aug. 21, 2002 (Working Draft for Internet Engineering Task Force), available at https://rip.psg.com/~randy/draft-ymbk-obscurity-01.html.

If a patch[10] is not yet available, the system owner can decide to take other measures, such as taking a system off-line or disabling the software, until a defense does become available.

Effects of Assumptions (A1), (A2), and (A3): In the Open Source paradigm, the costs of disclosure of a vulnerability are low because attackers learn little or nothing from the disclosure. The benefits of disclosure are high because of improved system design and actions taken by non-designers to protect their systems.

B. Case B: The Military Paradigm.

The assumptions in the military setting are directly contrary to the Open Source paradigm:

(B1) Attackers will learn a lot from disclosure of a vulnerability.
(B2) Disclosure will teach the designers little or nothing about how to improve the defenses.
(B3) Disclosure will prompt little or no improvement in defense by other defenders.

The intuition for assumption (B1) is that it is difficult in a military setting for the attackers to learn about a vulnerability. Consider a hill that is being defended by mines or camouflaged machine guns. Should the defenders publish the location of the defenses on the Internet? The answer clearly is no. It will be difficult and costly for the attackers to learn those locations and to determine the least-defended path up the hill. Colloquially and literally, the attackers will have to "pay in blood" to learn the weak points of the defense. Disclosure in this setting would help attackers considerably.

The intuition for assumption (B2) is a bit less clear-cut. It certainly is possible that public disclosure of a design will lead clever persons outside of the military to suggest improvements in design. More likely, however, the incremental learning from these outsiders will be modest at best. For specialized military topics, there is likely no pool of helpful outside experts comparable to Open Source programmers. Instead of depending on outsiders, the military will often hire or train the best available experts in specialized military equipment (tanks or fighter planes) or applications (battlefield communications). Public disclosure of the defenses will then do little to improve the design of the defenses.

Under assumption (B3), the military will often be the organization affected directly by a vulnerability. There may be counter-measures for land mines (magnetic detectors) or for camouflaged machine guns (infrared detectors). If so, then the military

[10] *See* Webopedia, *available at* http://webopedia.internet.com/TERM/p/patch.html (defining "patch": "Also called a service patch, a fix to a program bug. A patch is an actual piece of object code that is inserted into (patched into) an executable program. Patches typically are available as downloads over the Internet." *See also Understanding Patch and Update Management: Microsoft's Software Update Strategy*, Oct. 20, 2003, *available at* http://www.microsoft.com/security/whitepapers/patch)management.asp.

generally has confidential channels for telling its own people what to do in response. There are few or no third parties who would benefit from disclosure of the vulnerability or what to do about the vulnerability. (At least there are no third parties on "our side" that we want to tell.)

Turning briefly to the submarine situation during World War II, disclosure of the sailing time of a convoy helped attackers by revealing a vulnerability. Disclosure did little or nothing to help the Navy (the system designer for the defense) to protect the ships. Disclosure also did little to help other parties to defend themselves.[11] In this setting "loose lips" did indeed "sink ships" -- the costs of disclosure outweighed the benefits.

Effects of Assumptions B1, B2, and B3: In the military paradigm, the costs of disclosure of a vulnerability are high because attackers otherwise pay a high cost to learn of the vulnerability. The benefits of disclosure are low because outside designers are unlikely to improve the defenses and there are few or no third parties that the defenders wish to help through disclosure.

Taking the Open Source and military cases together, we can create a 2x2 matrix that visually shows the different effects of disclosure under the two paradigms. Under the Open Source assumptions, disclosure tends to improve the defense without helping the attackers. There are thus net benefits from disclosure. Under the military assumptions, the effects are reversed and there are net costs from disclosure.

	Help the Attackers Effect		
Help the Defenders Effect		**Low**	**High**
	High	**A: Open Source**	
	Low		**B: Military**

Table 1: Greater Disclosure Up and to the Left; Greater Secrecy Down and to the Right

[11] It is possible to imagine some assistance to third parties from disclosure. For instance, other ships might venture to sea if it becomes known that there is a convoy in another area that will draw the submarines' attacks. This benefit from disclosure, however, is likely to be outweighed by the harm to the convoy that becomes the target of the attack.

C. Case C: The Information Sharing Paradigm

The matrix also sheds light on when greater "information sharing" will improve security, such as the numerous information sharing provisions in the USA-PATRIOT Act[12] or proposals for the CIA and the FBI to share more of their data. Perhaps the easiest case to understand concerns the sharing of "watch lists" of suspected terrorists with defenders such as airport screeners, visa officers, and officials in other countries. Will greater disclosure of the watch list improve or harm security? The assumptions are:

(C1) Attackers may learn a lot from disclosure.
(C2) Disclosure may teach defenders how to design better systems.
(C3) Disclosure will allow more defenders to take protective actions.

The intuition for assumption (C1) is that broader dissemination of the watch list may tip off attackers who are on the list. The tip may occur either due to a mole (a rogue employee) or because the list is kept in an insecure place and gets leaked to the attackers. Persons who are on the list will then be on notice to avoid enforcement officials or to mask their identity. Persons who are not on the list will learn not to associate publicly with their colleagues on the watch list. Persons who are not on the list will also learn that they are "safe" and thus can fly on airplanes or otherwise get through screening processes. These "safe" people can then infiltrate defenses more effectively to spy or launch an attack.

The intuition for assumption (C2) is that broader use of watch lists, implemented by more defenders, may provide useful feedback for what sorts of watch lists are effective. A stronger intuition likely exists for assumption (C3). Putting the watch list into the hands of more defenders increases the likelihood of spotting and capturing the attacker. For instance, putting the picture of a "most wanted" criminal on television makes it harder for the criminal to escape. Especially where the criminal already knows that he or she is being chased, disclosure will help the defenders more than the criminal.

In practice, how the costs and benefits of disclosure compare will be an empirical question. Defenders will seek to create systems where defenders can effectively learn information while attackers cannot. As the number of defenders grows, however, it is less likely that every one of the defenders is trustworthy and every system containing the information is secure.[13] Information sharing is likely to have both costs and benefits, which will vary with the circumstances.

---

[12] *See* Peter P. Swire, "Information Sharing, the Patriot Act, and Privacy," (presentation made February 28, 2004), *available at* www.peterswire.net. USA-Patriot Act of 2001, Pub. L. No. 107-56, 115 Stat. 272, Secs. 203, 326.

[13] In some instances, technological measures may help get benefits of disclosure while minimizing the costs. The technological and institutional issues for doing so are beyond the scope of this paper. The most intense recent public debate has been about the CAPPS II system for screening airline passengers. *See, e.g.*, Center for Democracy and Technology, "TSA Issues Second Privacy Act Notice Expanding and Narrowing CAPPS II," July 31, 2003, *available at* http://www.cdt.org/wiretap.

Effects of Assumptions C1, C2, and C3: In the information sharing paradigm, there are significant costs and significant benefits from disclosure. The costs of disclosure may be high if attackers learn about the nature of the defense. The benefits of disclosure may be high if defenders can take additional, effective measures against the attackers.

D. Case D: The Public Domain

Another important possibility is that disclosure of a vulnerability will have low costs and low benefits. In some instances, a vulnerability is so minor that attackers will not be inclined to exploit it. More broadly, in many settings the information is already in the public domain – the relevant information is already available to interested attackers and defenders. In such settings, the assumptions are:

(D1) Attackers will learn little or nothing from the disclosure.
(D2) System designers will learn little or nothing from the disclosure.
(D3) Other defenders may learn little or a significant amount from the disclosure.

An example of information in the public domain is the street map for Manhattan or Washington, D.C. Having a detailed and accurate street map is a great advantage for an attacker. In war-time, attackers crave good maps as they move into enemy territory. Good maps allow precise planning, facilitate coordinated attacks, and reduce the risk of hidden features that can booby-trap the assault. In response, defenders who know the terrain may remove street signs or take other measures to prevent the attackers from learning the area.

As part of the war on terrorism, it might thus be tempting for the United States to try to prevent terrorists from getting accurate street maps of potential targets such as Manhattan or Washington, D.C. The problem, however, is obvious. Detailed and accurate street maps of those cities are in the public domain, with innumerable copies in print and on the Internet. It would be very expensive even to try to hide the maps and such efforts would almost certainly be futile. In addition to these costs of trying to hide the maps, there would be substantial costs to all the legitimate users of the maps.

In terms of the three assumptions, assumption (D1) is that attackers would learn little or nothing new from a "disclosure" such as the publishing of an additional street map. Assumption (D2) is that the designers of the defense would learn little or nothing when a new street map is published. Assumption (D3) is that a new street map may in fact be of some use to other "defenders" – legitimate users of the information including tourists, urban planners, and all others who rely on street maps.

Looked at from the other direction, efforts to hide or "re-classify" information will often be expensive and not very effective in an era of the Internet, on-line search

engines, and archiving of information once it has been on the Internet.[14] The benefits of trying to hide the information will often be small because determined attackers will still have the information. The costs of trying to hide the information may be considerable, both in the effort to find and destroy copies that already exist and in the effect on legitimate users of the information.[15] Once a secret is exposed, it is often costly or impossible to put the genie back in the bottle.

Effects of Assumptions (D1), (D2), and (D3): For information in the public domain, there are few or no costs from additional disclosure. There may be benefits from additional disclosure if additional legitimate users (defenders) learn from the disclosure. There are likely high costs from trying to hide data once it is in the public domain.

E. The 2x2 Matrix for When Disclosure Improves Security

With the addition of Case C on Information Sharing and Case D on the Public Domain, each cell of the 2x2 matrix has been filled in. Table 2 shows the result:

		Help the Attackers Effect	
		Low	**High**
Help the Defenders Effect	**High**	**A: Open Source**	**C. Information Sharing**
	Low	**D. Public Domain**	**B: Military**

Table 2: Greater Disclosure Up and to the Left;
Greater Secrecy Down and to the Right

At this stage, a few comments will help to emphasize what is and is not accomplished by Table 2. First, a chief goal of the table is to organize our current thinking about the dueling approaches of disclosure ("no security through obscurity") and

[14] For one informative discussion of the wealth of information available through the Google service, see Scott Granneman, "The Perils of Googling," The Register, Mar. 10, 2004, *available at* http://www.theregister.co.uk/content/55/36142.html.

[15] The discussion here focuses only on the extent to which the disclosure will help or hinder the attackers. Efforts to censor information in the public domain also can obviously raise serious First Amendment and other problems. Eugene Volokh has written an excellent analysis of these issues, in an approach that is congruent in a number of respects with the analysis in this paper. Eugene Volokh, *Crime-Facilitating Speech*, (unpublished draft, 2004).

secrecy ("loose lips sink ships"). By clarifying the assumptions underlying those two scenarios, the table also reveals the assumptions underlying two other common scenarios – information sharing and the public domain. Second, the table simplifies reality by showing a binary split between high and low effects of helping the attackers and improving the defense. In reality, there is a continuum between high and low effects. Real-world examples will range along the two dimensions. Third, the table is based on *assumptions* about the effects of disclosure on attackers and defenders. Conclusions about the desirability about a disclosure will depend on how valid the assumptions are in a given setting.

II. The Key Reasons Computer and Network Security May Vary from Other Security Problems

In the legal academy, there has been a lively debate about the extent to which cyberspace (and the law of cyberspace) is different from the physical world (and the law of the physical world). For instance, writers such as David Post and David Johnson have stressed the uniqueness of the Internet, while writers such as Frank Easterbrook and Jack Goldsmith have stressed how the law of the Internet is fundamentally similar to previous legal issues.[16] The topic of this Part is to examine the extent and nature of the differences between computer and network security, on the one hand, and the military and other traditional security problems of the physical world, on the other.

The conclusion here is that there is no *logical* or *necessary* difference between cybersecurity and physical security. One can generate examples where the nature of the security challenge and the optimal degree of disclosure are the same regardless of what is being protected. Nonetheless, the claim here is that there are reasons why there are *commonly* important differences between cybersecurity and physical security. These differences, I believe, contribute a great deal to why so many cybersecurity experts intuitively believe in "no security through obscurity" while so many military and other physical security experts intuitively believe that "loose lips sink ships."

A. Hiddenness and the First-Time Attack

Here is an organizing concept for when hiddenness helps security: a hidden feature is more likely to be effective against the first attack, but less likely to be effective against repeated attacks. Consider an example from the physical world. A fort is protected by a simple security device that relies on hiddenness. On the path up to the fort there is a pit covered by leaves, with a sharpened stick at the bottom. The first time an attacker comes up the path, the attacker might fall into the pit. Even if hiddenness works against the first attacker, however, later attackers will likely not "fall" for the same trick. The later attackers may know where the pit is, or they may come equipped with sticks that probe the path so that they don't fall in. In this simple example, using obscurity may

[16] *See* Jack L. Goldsmith, "Against Cyberanarchy," 65 U. Chi. L. Rev. 1199, 1199 n.3 (1998) (collecting citations to works of Post, Johnson, and others who stress uniqueness of cyberspace law); Frank H. Easterbrook, "Cyberspace and the Law of the Horse," 1996 U. Chi. L. Forum 207.

work against the first attacker but is unlikely to work once the attackers learn to watch for the hidden pit.

The concept of the first-time attack can be generalized. Consider a "hidden" defensive feature as one that is not known initially to any attacker. The effectiveness of hiddenness will be a function of five variables:

(1) The effectiveness of the defensive feature at stopping the first attack. ("***E***" for effectiveness.)

(2) The number of attacks. ("***N***" for number of attacks.)

(3) The extent to which an attacker learns from previous attacks. ("***L***" for the learning that occurs.)

(4) The extent to which the attacker communicates this learning to other attackers. ("***C***" for communication.)

(5) The extent to which the defenders can effectively alter the defensive feature before the next attack. ("***A***" for alteration of the defense.). Note that the alteration may come from the system designer/defender (**A-D)**. The proposed alteration may also come from Third Parties who learn how to fix the vulnerability (**A-T)**, such as when an Open Source programmer designs a patch.

The effectiveness of hiddenness will vary directly with greater initial effectiveness (***E***) and greater ability by the designer to alter the defense (***A-D***). It will vary inversely with the number of attacks (***N***), the degree of learning by attackers (***L***), the ability of attackers to communicate (***C***), and the ability of third parties to alter the defense (***A-T***). When the effects of ***N***, ***L***, and ***C*** grow very large, there will be no usefulness of hiding the defensive feature. All attackers will then know everything about the "hidden" feature.[17]

The military and Open Source examples explained earlier illustrate how the effectiveness of hiddenness can vary depending on the five variables. Start with the example of camouflaged machine guns guarding a hill, where the effect of hiddenness is the difference between announcing the location of the machine guns and keeping them hidden. Initially, the attackers do not know where the machine guns are hidden. ***E***, the effectiveness of the defense, will likely be high against infantry attackers because the

[17] The discussion here does not present a detailed mathematical model of how hiddenness contributes to security. Identification of the five variables, however, should enable those who are mathematically more skilled than I am to build such a model. As suggested in conversation by Rena Mears, the approach here implicitly assumes a calculus function where the effectiveness of hiddenness goes to zero as the number of attacks approaches infinity (assuming a positive value for ***L*** and ***C***) (and also assuming the effect of ***L*** and ***C*** in helping attackers outweighs the effect of alterations in helping defenders)..

hidden guns will make it hard for attackers to find a safe path up the hill.[18] ***N***, the number of attacks, will be low. Each attack is a major event that is costly in terms of casualties. ***L***, or learning, will vary depending on the ability of an individual attacker to get back safely from the first attack or go around the machine gun nest. If all the attackers are killed in the attack, then ***L*** will be zero. ***C***, or the ability to communicate, will vary depending on the ability of any individual attacker to tell the rest of the troops about the location of the hidden guns. If the attackers have radios, then there will be a high ***C*** because they can tell their comrades what locations to avoid. If the attackers have to rely on word-of-mouth, then ***C*** will be low and the defense may have time to set up a new ambush in time for the second attack.

Pulling these observations together, each attack on the hidden machine guns is very expensive for the attackers. Hiddenness benefits the defender in the first attack. The number of attacks will be small (would there be even three or four charges against a well-defended hill?). Attackers may not learn quickly about the hidden defenses, may find it difficult to communicate their learning to the other attackers, and may face a changed defense by the time they launch their next attack. For all of these reasons, hiddenness will benefit the defense.

Under the assumptions used thus far for Open Source software, hiddenness will be much less effective. It is possible that the initial effectiveness of a defensive trick, ***E***, will be substantial. The number of attacks, ***N***, will quite possibly be high. Malicious hackers can probe for weaknesses in a software product over and over again. The attackers learn (***L***) from the attacks, such as by seeing whether they can gain control over the software. Attackers can communicate (***C***) about flaws, such as by posting their exploits to web sites to let other attackers know about the flaws.

Under these assumptions, each attack on a software program is very cheap – attackers can probe the program over and over again from the comfort of their own homes or computer labs. They learn about flaws and tell others about flaws. Very quickly, under these assumptions, the hidden defense is exposed to the world. Thus, there is "no security through obscurity."

The possibility of altering the defense also works differently than for the physical attack against machine guns. In the machine gun setting, the defense may be able to move the guns between each attack. If that is true, then the second ambush may be as effective as the first, and hiddenness once more favors the defender. Under the Open Source assumptions, disclosure of the vulnerability actually increases ***A***, the likelihood of effective alteration of the defense. The idea is that other Open Source programmers will come forward to write a patch for the vulnerability. In terms of hiddenness, improved protection against the next attack works in opposite ways for the machine gun and Open Source examples.

[18] The example here assumes foot soldiers charging up a hill against machine guns. If the attack is made by heavy tanks, then ordinary machine guns will not stop the attack. For the tank attack, the value of ***E***, the initial effectiveness, would be low.

B. Uniqueness of the Defense

How should we refer to the effect of the five variables? "First-time attack" has the advantage of communicating to a wide audience. Using ordinary English, a reader can grasp the idea that a hidden trick may work against the first attack but fail against the 1000th attack. The problem with the term "first-time attack," however, is generalizing the effect to "second-time attacks" (hiddenness may still work very well), "twentieth-time attacks" (hard to know how well hiddenness will work), and "nth-time attacks" (the hidden features will quite possibly be discovered).

This paper will use the word "uniqueness" to refer to the usefulness of hiddenness for the defense. Despite the possible complaints of English teachers,[19] this paper will discuss uniqueness as a function, varying from "unique" or "entirely unique" down through "somewhat unique" to "not unique at all."[20] The function for uniqueness (***U***), or the usefulness of hiddenness for the defense, is thus:

$$\boldsymbol{U} = \mathrm{f}\,(\boldsymbol{E, N, L, C, A})$$

Under the terminology employed here, "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 defense (***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.

C. Why Low Uniqueness May Be Common for Computer and Network Security

Important areas of computer and network security include: perimeter defense such as firewalls; mass-market software, including video games; and encryption. For each of these areas there will often be a low degree of uniqueness, so secrecy is unlikely to be very effective.

1. Firewalls. There is a plausible case that firewalls are subject to a large number of attacks (***N***), considerable learning by attackers (***L***), and effective communications among attackers (***C***). Using the Internet, attackers can probe a firewall from anywhere on the planet. They can attack again and again at low cost, trying various combinations of attacks until they find one that works. They can then tell other attackers about the vulnerability, such as by posting a script of the attack to a web site or e-mail list. Even

[19] One web page lists "errors" in English usage, and says: "'Unique' singles out one of a kind. That 'un' at the beginning is a form of 'one.' A thing is unique (the only one of its kind) or it is not. Something may be almost unique (there are very few like it), but nothing is "very unique." http://www.wsu.edu:8080/~brians/errors/unique.html.

[20] When I gave this paper at a conference at the Stanford Law School, Bruce Schneier and Matt Blaze both suggested the term "instance" to refer to what I am here calling "uniqueness." I have chosen the latter term for two main reasons. First, "instance" has so many uses in English that it may be confusing to readers for it to have a more technical definition. Second, my sense is that readers will intuitively understand the idea of different degrees of uniqueness.

unskilled "script kiddies"[21] may then be able to use the attack to pierce that firewall or other firewalls that use the same defenses.

Comparison with an attack on a walled city illuminates the way that computer and physical attacks are both similar and different. The similarities between a computer firewall and a medieval city wall are easy to see. A strong barrier is designed to allow friends to enter but keep foes out. Either sort of defense can be set to various levels of security. In times of peace, a city gate may allow anyone to enter, with guards on hand to handle anyone suspicious. At a higher level of alert, guards might check the credentials of each person before entering the city. During a siege, the gates might be closed completely, barring all entry. Additional security might exist within the city wall. For instance, the armory (containing weapons), the mint (containing treasure), and the castle keep (containing the ruler) all would have additional protections against entry.

A company's firewall is similar. For non-essential systems most messages will be allowed entry. For secure systems, a password or other credential is required. Under severe conditions, such as a distributed denial of service attack, all messages may be blocked from entering the company's system. Additional security will exist for priority functions, such as the system security (the armory), the corporate treasury (the mint), and the root directory (the ruler's residence).

Along with these similarities, it is logically possible for attacks against a physical wall to have high ***N***, ***L***, and ***C***. For a long and badly defended wall, for instance, intruders might repeatedly probe for weak spots, learn about weak spots, and tell fellow attackers where to enter.[22]

Many attacks against a city wall, however, do not fit that pattern. In medieval warfare, an attack against a walled city was a major event in which many people might die. Any hidden trick by the defenders might cost attackers' lives or save defenders' lives before the attackers learned how to counter the trick. The number of attacks was low, attackers might not survive to tell about weak spots, and communication back to the attacking generals was rudimentary. Similarly, any hidden weaknesses might not be revealed in time to help the attack. In short, ***N***, ***L***, and ***C*** would all be low.

In sum, low levels of *N*, *L*, and *C* likely meant that medieval city walls had high uniqueness – secrecy was likely to be a useful tool. Firewalls using standard software likely have low uniqueness due to the high levels of *N*, *L*, and *C*.[23]

---

[21] "Script kiddies" are unskilled programmers who merely follow a script rather than understanding how to write code themselves. *See, e.g.*, The Jargon Dictionary, *available at* http://info.astrian.net/jargon/terms/s/script_kiddies.html (defining script kiddies as "the lowest form of cracker; script kiddies do mischief with scripts and programs written by others, often without understanding the exploit.")

[22] An example of a physical barrier with high ***N***, ***L***, and ***C*** might be the United States border with Mexico. There are many persons who seek to cross the border, there are professionals who learn the soft spots in the defenses, and others who wish to cross the border learn from earlier successes.

[23] Despite the intuition that firewalls have low uniqueness, I have talked with some computer security experts who build higher uniqueness into their own firewalls. Even for some experts who support the idea

2. Mass-market software and computer games. A next major topic of modern computer security is how to protect standardized software against hackers. Popular products may be on thousands or millions of desktops. Designers of standardized software might try to use hiddenness to stop the hackers. For instance, the designer might have a program freeze up permanently if a user hacked into inappropriate portions of the software. This kind of defense would be similar to falling into the pit covered with leaves – the attackers who goes into the wrong place never comes out again.

This hiddenness will often not work well, however, for mass-market software. Suppose, for instance, that there are a dozen paths for hacking a piece of code to do something forbidden such as send a virus or make illegal copies. Suppose the designer puts traps on eleven of the twelve, to freeze up the program permanently if a hacker trespasses into the wrong part of the code. Suppose further that the designer leaves the twelfth path free so that the designer can get back in to rewrite the code.

This sort of defense would work reasonably well against a one-time attack. In the physical world, an attacker would face a grave risk (11 out of 12) of falling into the pit and getting injured. Similarly, in the computer world, a hacker who can get only one copy of the program, and who needs that program to keep functioning, will find it too risky to fool around with the program and likely have it freeze into uselessness. By contrast, a hacker who can buy (or illegally copy) many copies of the program will not mind much if a few copies freeze up. This hacker can systematically try one possible attacker after another until something works – a high ***N*** and ***L***. Meanwhile, other hackers around the world also try their favorite attacks, and the hackers can communicate amongst themselves when they find a vulnerability – a high ***C***.

The combination of high ***N, L***, and ***C*** also exist for computer and video games today when players try to "beat the game."[24] "Beating the game" is a (presumably) innocent version of hacking a software system – users ultimately reach their goal of gaining control over the software. An old-fashioned (although perhaps satisfying) way to "beat the game" is to keep trying by yourself until you overcome all the obstacles. As an alternative, video game players today can also enlist a global network of fellow aficionados. Web sites appear almost instantly after release of a game. The sites offer "secrets" (press the third brick on the left to get a magic sword), "walk throughs" (on Level 13 here are the seven things you have to do before you attack the dragon), and even "cheats" (if you enter this code, your player will become invulnerable to all attacks and as strong as Superman). Translated back into the language of computer security, there is a high number of attacks, ***N*** – just ask the parents. Users learn from experience and communicate that learning – a high ***L*** and ***C***. A hidden measure by the game designers will not stay hidden for long.

---

of "no security through obscurity" there is an understanding that putting some hidden tricks into a defensive system such as a firewall can be helpful. The hidden or subtle changes notably can stop attacks by "script kiddies" and others who are not able to modify their attacks in the face of a new defense.

[24] This paragraph is based on insights from my sons Nathan and Jesse Swire, now 15 and 13.

In summary, where there are high levels of *N*, *L*, and *C* for attacks on mass-market software, there will tend to be low uniqueness and little "security through obscurity."

3. Encryption. Encryption is a third major area of modern computer security, along with system defense (firewalls) and defending software. The word "encryption" comes from the Greek word for "hidden," so it might seem exceedingly odd to say that being hidden does not work well for encryption.[25] Yet, in the sense used in this article, that is precisely the claim. The question, for our purposes, is whether hiddenness paired with encryption that suffers from vulnerabilities will succeed, or whether instead security can be provided only by strong encryption, *i.e.*, encryption that is successful even when the attacker knows the method used to encrypt the message.

Modern cryptographers are likely the most avid believers that there is no security through obscurity. Cryptographic authority Bruce Schneier has stated:

> A basic rule of cryptography is to use published, public, algorithms and protocols. This principle was first stated in 1883 by Auguste Kerckhoffs: in a well-designed cryptographic system, only the key needs to be secret; there should be no secrecy in the algorithm. Modern cryptographers have embraced this principle, calling anything else "security by obscurity." Any system that tries to keep its algorithms secret for security reasons is quickly dismissed by the community, and referred to as "snake oil" or even worse.[26]

The Schneier quote, with its discussion of "snake oil," highlights the risk that a vendor will dupe purchasers of an allegedly secure system. Once the system is exposed to attack, however, the system may have only weak protections, and all of the communications of the purchaser may thus be exposed to view. Having "published, public, algorithms and protocols" is thus an important consumer protection against the vendor who tries to hide the vulnerabilities of a weak system.

A second reason for the cryptographers' belief in openness is that a secret is unlikely to remain secret when known to a large number of people. Cryptography today is used by an enormous number of users on the Internet. In earlier times, by contrast, encryption was used by far fewer persons, most prominently by diplomats and the military. Encryption became more widespread when people wished to send a lot of important messages through a channel where other people could see or hear the message. In times when the post was not secure, letter writers used encryption. In the days of the

---

[25] For excellent historical introductions to encryption, see DAVID KAHN, THE CODEBREAKERS: THE STORY OF SECRET WRITING (1996); SIMON SINGH, THE CODE BOOK: THE EVOLUTION OF SECRECY FROM MARY QUEEN OF SCOTS TO QUANTUM CRYPTOGRAPHY (1999).

[26] Bruce Schneier, "Secrecy, Security, and Obscurity," Cryptogram Newsletter, May 15, 2002, *available at* http://www.schneier.com/crypto-gram-0205.html. Schneier returned to these issues in Beyond Fear: Thinking Sensibly about Security in an Uncertain World 126-32 (2003). These writings are extremely insightful and the most detailed I have found on the issues that appear both in this article and in the draft version of my work that was posted to the Internet in 2001, *see supra* note *.

telegraph, many businesses used encryption to keep their commercial secrets away from the eyes of the telegraph operators. For radio communications, anyone with a receiver could hear the message. Most famously, German submarines in World War II used the Enigma system when radioing back to headquarters. Allied cryptographers learned to break the system after enormous effort, helping to win the war and more or less inventing the computer as a by-product.

The need for encryption is thus not new with the Internet. But the Internet has been accompanied by an enormous increase in the need for and use of encryption by ordinary people and businesses. The Internet is a famously "open" system.[27] A message from Alice to Bob is typically routed through many computers on its way through the Internet. A hacker might be in control of any of those computers. The hacker might make a copy of all the messages coming through the system and then comb through the messages looking for any that have commercial, diplomatic, or military value. In response, Alice and Bob need to encrypt their important messages, containing credit card numbers, trade secrets, large transfers of currency, and anything else they don't want the hacker to read and copy.

The Internet does more than increase the number of messages that use encryption. The Internet has also accelerated demand for public-key encryption approaches that permit anyone to send an encrypted message to anyone else. The basic idea of a public-key system is that a user, Alice, can send a message to a recipient, Bob, whom she has never met before.[28] She uses Bob's public key to encrypt her message. Bob can then decrypt it using his private key. The public key can be posted on the Internet or otherwise revealed to the world. The private key is kept secret by Bob and not made known to attackers. The combination of many messages through insecure channels (the Internet) and many users who wish to communicate securely with each other (as in E-commerce) has meant that an unprecedented number of individuals rely on cryptosystems[29] that are widely deployed.

---

[27] *See generally* Jane Kaufman Winn, O*pen Systems, Free Markets, and Regulation of Internet Commerce*, 72 TUL. L. REV. 1177 (1998) (discussing the openness of the Internet).

[28] For further discussion, *see, e.g.*, Bruce Schneier, Applied Cryptography ch. 19 (2d ed. 1996).

[29] Modern encryption draws a distinction between the "cryptosystem" and the "key." The cryptosystem is a mathematical technique that has a standard way to re-arrange symbols ("put every second letter in front of the letter before it") and substitute one symbol for another ("change each letter A into the number 1"). The most widely-used modern cryptosystems publish the algorithm for converting between plaintext (readable English) and ciphertext (the message as transmitted in its encrypted form). A well-known example is the RSA algorithm developed in 1978 by mathematicians Ronald Rivest, Avi Shamir, and Leonard Adleman. The security of the RSA cryptosystem depends on a mathematical algorithm that is easy to calculate in one direction (when one encrypts the message) but extremely difficult to calculate in the other direction (when an unauthorized person tries to decrypt the message.) For the mathematical basis of the RSA algorithm, created in 1978, see "What is the RSA Cryptosystem?", available at http://www.rsasecurity.com/rsalabs/node.asp?id=2214.

The security of the RSA cryptosystem also depends on each user having a secret key to turn ciphertext back into plaintext. The idea of a key is simple enough. Suppose that the cryptosystem turns each letter into a number, such as A=1, B=2, C=3, and so on. There are 26 possible starting points, such as A=25, B=26, C=1, and so on. In this simplified example, the cryptosystem is a regular pattern for turning letters into numbers. The key is knowing how to begin the calculation, by knowing which number corresponds to the letter A. In actual cryptosystems, the key is a long chain of randomized numbers.

Given this understanding of today's networked encryption, we can now better understand why modern cryptographers believe there is no security through obscurity. Because so many communications flow through the Internet and can be read by hackers, the number of attacks, ***N***, is extremely high. If ***L*** and ***C*** are even slightly positive, then attackers will learn about the vulnerabilities in a method for encrypting messages and communicate about those vulnerabilities to others. The response by cryptographers is to use methods for encryption that do not rely on secrecy. Instead, cryptographers increase the length of the secret key to try to make brute force attacks prohibitively costly.

The combined effects of ***N, L***, and ***C*** mean that the cost of disclosure of the cryptosystem – the help-the-attackers effect – is low. The benefit of disclosure to defenders is also likely to be high. For one thing, use of a public-key algorithm means that myriad users can easily send encrypted messages to each other. In addition, there is likely a high value for ***A***, the ability of defenders to improve the defensive system. The rise of the Internet and the spread of public-key encryption means that the number of encryption experts has grown rapidly in recent years. The likelihood of improved defenses is thus substantial: "The long history of cryptography and cryptanalysis has shown time and time again that open discussion and analysis of algorithms exposes weaknesses not thought of by the original authors, and thereby leads to better and more secure algorithms."[30]

Before leaving the topic of encryption, it might be useful to see how this conclusion – the advantage of an open cryptosystem – would have been less true in Roman or Medieval times. In that setting, there likely would have been lower ***N, L***, ***C***, and ***A***. The number of encrypted messages subject to interception would have been far lower than on the Internet. The sophistication of those intercepting the messages would have been lower. Slow communications would have meant that other attackers would have learned very slowly if at all from the breakthrough by one attacker. In addition, the chances of "outside cryptographic experts" improving the system would have been low. All of these variables would therefore have pointed toward the usefulness of a hidden cryptosystem, in contrast to conditions today.[31]

Attackers who do not have the key then need to try every possible combination of numbers until a key fits the lock (decrypts this plaintext). Trying each of the combinations, which can easily number in the billions, trillions, and up, is called a "brute force attack." An attacker who can try every single possible key will eventually be able to read the code. The response by those who build cryptosystems is to try to make the number of combinations so large that no available computer can try all the combinations.

[30] Randy Bush & Steven Bellovin, "Security Through Obscurity Dangerous," Internet Society Working Draft, Aug. 21, 2002, *available at* https://rip.psg.com/~randy/draft-ymbk-obscurity-01.html

[31] In comments on an earlier draft, cryptographer Susan Landau disagreed with the discussion of the role of hiddenness in earlier times. She mentioned a 14th Century Arabic encyclopedia, the Subh al-a 'sha, that contained sophisticated mathematical techniques for breaking ciphers. In response, the claim here is that secrecy is more likely to have net benefits in situations with lower ***N, L***, ***C***, and ***A***. Where attackers such as users of that encyclopedia have sophisticated techniques, they will have higher ***L***, reducing the effectiveness of secrecy. The claim in the text is that earlier periods generally had far lower ***N, L***, and ***C*** than would attacks today on a widely-used cryptosystem on the Internet. Modern attackers will thus be more efficient at overcoming hidden defenses (due to today's higher learning) and modern defenders will be more likely to get suggestions for useful alterations (due to today's larger group of potentially helpful

In summary, secrecy in modern cryptosystems is unlikely to be useful due to high *N*, *L*, *C*, and ***A*. Modern encryption relies, however, on strict secrecy for private keys.**

III. Relaxing the Open Source Assumptions – Computer and Network Security in the Real World

Part II sought to explain why computer security experts so often believe there is no security through obscurity. Firewalls, mass-market software, and encryption are major topics for computer and network security. In each setting, there are typically high values for number of attacks (***N***), learning by attackers (***L***), and communication among attackers (***C***). Secrecy is of relatively little use in settings with high ***N***, ***L***, and ***C*** – attackers will soon learn about the hidden tricks. By contrast, many physical-world security settings have lower values for ***N***, ***L***, and ***C***. In these settings of persistent and higher uniqueness, secrecy is of greater value to the defense.

Part II thus solidifies the assumptions of the Open Source paradigm, that (1) disclosure will offer little or no help to attackers; (2) disclosure will tend to upgrade the design of defenses; and (3) disclosure will spread effective defenses to third parties. High levels of ***N***, ***L***, and ***C*** strengthen the first assumption, because attackers will quickly learn about secrets. Alterations (***A***) from outside experts, in cryptosystems and elsewhere, fit with the second assumption. Finally, high levels of ***A*** and ***C*** will alert other defenders to vulnerabilities under the third assumption.

After this reinforcement of the Open Source assumptions, Part III will now try to test the assumptions against the real world. In practice, secrecy will often be of greater use than suggested by the assumptions of the Open Source paradigm.

A. The Assumption that Disclosure Will Not Help the Attackers

The first assumption in the Open Source paradigm is that disclosure will provide little or no help to the attackers. The assumption is that there are many capable persons who are willing and able to launch attacks against firewalls, mass-market software, and cryptosystems.

To scrutinize this assumption, it is important first to develop the intuition that the public domain of information is expanding in a world of search engines such as Google. Next, disclosure can sometimes help defenders when the disclosure deters attacks. Third, the case for disclosure of private keys, such as cryptographic keys, is especially weak. Fourth, the area of surveillance is subject to a different analysis. Finally, the discussion turns to a more specific discussion of the extent to which attackers already know about how to launch effective attacks against firewalls, mass-market software, and cryptosystems.

---

cryptographic experts). There will thus be higher expected benefits today of disclosure of the cryptosystem.

1. The Enlargement of the Public Domain in a World of Search Engines. Do attackers know specific facts about defenders? The answer today, in a world of the Internet and search engines, is that the cost of doing searches has gone way down. Many facts that were impossible or costly to find in the past are easy to find today.

All readers of this article know this to some extent, but it is helpful to flesh out some of the reasons that so much more information is today in the public domain. The Internet itself has only recently expanded beyond the domain of DARPA[32] and the academic community. Indeed, it was not until 1992 that the terms of service for the Internet changed to permit commercial activity on the Internet.[33] The growth in commercial activity coincided with the incredible expansion of Internet usage, so that ordinary people all over the world could find out information, at no cost, about a huge range of topics. Search engines have made it trivially easy to search through the many web sites to find specific information. Google was launched in 1998 and indexed 30 million items at that time. Today, it indexes over 6 billion items.[34]

At a simple yet powerful level, the ubiquity of search engines (and the other research tools of the Information Age) increases the knowledge available to attackers. Attackers can correlate information from diverse sources to infer facts that are themselves not explicitly made public. Attackers can communicate with other attackers through blogs[35], web sites, and global, free e-mail. Search engines are extremely useful. For instance, you can pick the mass-market software or firewall you wish to attack and search on Google for the name of the product and "bugs" or "vulnerabilities." If you do so, you may find a patch that has already been announced for the known bug. In launching actual attacks, you are also likely to discover that a large portion of the product's users have not installed the patch.

The increase of information in the public domain increases the set of instances where the Open Source paradigm is a better approximation than the military paradigm. More often than before, disclosure of a security flaw will add little or nothing to attackers' knowledge. It will be harder to keep many things secret, because attackers will be able to infer the truth from other available information. At the very least, the ubiquity

---

[32] DARPA is the Defense Advanced Research Projects Agency, which played a key role in fostering the early stages of the Internet. *See* Michael Hauben, "History of ARPANET; Behind the Net - The untold history of the ARPANET; Or - The "Open" History of the ARPANET/Internet," *available at* http://www.dei.isep.ipp.pt/docs/arpa.html.

[33] The Scientific and Advanced Technology Act of 1992, signed into law on October 23, 1992, "subtly modified [the National Science Foundation's] authority to support computer networks that are not limited to research and education." NATIONAL SCIENCE FOUNDATION, OFFICE OF INSPECTOR GENERAL, REVIEW OF NSFNET, March 23, 1993 (citing 42 U.S.C. § 1862(g)). This change was one important legal step toward development of commercial activity over what is now called the Internet.

[34] Robert Weisman, "Investors Monitoring Climate for Google IPO," Miami-Herald.com, Mar. 21, 2004, *available at* http://www.miami.com/mld/miamiherald/business/national/8243019.htm.

[35] For an early discussion of the legal implications of weblogs, or "blogs,", see Attiya Malik, *Are You Content with the Content? Intellectual Property Implications of Weblog Publishing*, 21 John Marshall J. Computer & Information L.439. (2003).

of search engines increases the costs of trying to keep information out of the hands of attackers.[36]

In summary, the growth of the Internet and of search engines means that the optimal solution often shifts toward openness in weighing the costs and benefits of disclosure. In many instances, the help-the-attacker effect is likely to be low, while the costs to defenders of trying to keep secrets will have risen.

2. Deterrence as a Result of Disclosure. Up until this point, the focus has been on how disclosure may reveal vulnerabilities and thus help the attackers. More generally, the analysis should include the full range of ways that attackers might respond to disclosure about the defense. One principal way that disclosure can help the defense is through deterrence – the effect of disclosure on *reducing* the likelihood of attack.

The distinction between "strong" and "weak" is likely the main axis for when disclosure will create deterrence. Attackers who see a "strong" defense will tend to be less likely to attack. Attackers who see a "weak" defense are more likely to believe that they will be able to overcome the defenses. This effect is not always true – a "strong" defense, for instance, might be a clue to an attacker that something valuable is contained inside.[37] Nonetheless, the appearance of a "strong" defense is generally a good predictor for the magnitude of deterrence.[38]

---

[36] There is a growing literature that laments the shrinking of the public domain. *E.g.*, Lawrence Lessig, The Future of Ideas: The Fate of the Commons in a Connected World (2002); James Boyle, "The Second Enclosure Movement and the Construction of the Public Domain," 66 L. & Contemp. Prob. 33 (2003). This literature emphasizes, for instance, the ways in which copyright and other intellectual property rules have expanded. Even though copyright law itself does not apply to facts, some actual and proposed legal developments could reduce the set of facts available to the public. For instance, the anti-circumvention provisions of the Digital Millennium Copyright Act, 42 U.S.C. § 1201, can make it illegal to access facts that are in a format protected by anti-circumvention measures. In addition, there have been repeated legislative attempts in the United States to enact *sui generis* database protection, which would create new limits on the ability of users to reproduce facts in certain databases. Jonathan Band, "New Theories of Database Protection," Managing Intellectual Property (2003).

Notwithstanding these concerns, the argument here is that the development of the Internet and of search engines has made available an increased range of factual information at lower cost than previously. Especially in the wake of the attacks of September 11, there have been some measures by the U.S. government to reduce the information available to the public. Edward Lee, "The Public's Domain: The Evolution of Legal Restraints on the Government's Power to Control Public Access Through Secrecy or Intellectual Property," 55 Hastings L.J. 91 (2003). Despite these changes, vastly more security information is available today to a teenage hacker or a foreign terrorist than would have been true before the rise of the Internet.

[37] As another example where deterrence would not succeed, some attackers might be attracted to a strongly defended target simply because it is strongly defended. Just as medieval knights sought glory by attacking famous champions, modern-day hackers sometimes seek "hacker glory" by attacking systems that are thought to be highly secure.

[38] The "strong"/"weak" distinction was first suggested to me by Jim Steinberg, who served as Deputy National Security Advisor under President Clinton. The fact that the suggestion came from a person steeped in national security issues suggests that the deterrence effect may implicitly be an important way that military and national security experts decide when disclosure will help security.

Deterrence can exist because the defense is strong in an absolute sense. In such circumstances, the defender will perceive that the costs of the attack are greater than the benefits. An example in the physical world is if there is a high fence, topped with razor wire and with surveillance cameras in clear sight. A potential trespasser who sees this defense may estimate that it will be difficult to climb the fence, dangerous to get over the razor wire, and risky in terms of being detected and caught.

Deterrence can also exist in a relative sense. There is the old story about the two hikers in the woods who see a dangerous bear rushing toward them. One of the hikers turns around and starts running. The other hiker asks why he is running when everyone knows that bears can run faster than people. The first hiker responds: "I don't have to run faster than the bear. I just have to run faster than you." In terms of deterrence, a house with bars on the windows and large locks on the front door may simply be more trouble to attack than a neighboring house that lacks these features. The visible defense measures, in such circumstances, may shift the risk of attack to the neighboring house.

An essential element to successful deterrence is that the attackers know about the strong defense. This element was memorably missing in the movie "Dr. Strangelove," where the Soviet Union failed to tell the rest of the world about the existence of a doomsday device that would be triggered by any nuclear attack. When one nuclear bomb was accidentally used, the entire world was destroyed. The complete failure of communication in that instance drives home the point – it is the perception of the defense by the attackers that is key to deterrence.

In summary, the effects of disclosure on security include the deterrent effect on attacks (a help-the-defense effect) as well as the help-the-attackers effect discussed previously. The chief predictor of deterrence is the extent to which attackers perceive the defense as strong.

3. Don't Disclose Private Keys, Passwords, or Combinations to a Safe. The discussion of encryption, above, drew a sharp distinction between the cryptosystem and the private key. Modern cryptographers generally support "no security through obscurity" and favor disclosure of the cryptosystem. They also support secrecy for the private key or password. Modern cryptographic systems feature a high initial effectiveness for the cryptosystem (***E***). They also are resistant to a high number of attacks (***N***). The private keys are long enough to require brute force attacks that are too lengthy for attackers to undertake.

A similar analysis applies to physical protections such as a combination safe. The initial effectiveness (***E***) is high because attackers cannot easily get through the metal skin of the safe. The combination of the safe is then complicated enough to make a brute force attack difficult (resist against a high ***N***). A complex combination can be very effective -- bank robbers typically do not wish to stay in the bank vault long enough to try every possible combination to open the safe.

For passwords, good practice is to make it difficult for attackers to guess the password. Programs to guess passwords are easily available on the Internet.[39] In response, good practice is to require a password to include symbols and numbers in addition to letters. That practice increases the initial effectiveness (***E***) by forcing users not to use the defaults that come with software or common terms such as "password." Use of different characters in the password increases the number of attacks (***N***) needed to guess the password. In addition, altering the password periodically (***A***) reduces the likelihood that attackers can continue to take advantage of one successful attack.

For all three defenses – the private key, the combination to the safe, and the password – there is a large help-the-attacker effect from disclosure of the secret information. All three defenses are designed to frustrate a brute force attack by having too many possible combinations. If there is disclosure of the secret key, then the entire defensive strategy falls apart.

Is there a help-the-defender effect from disclosure? Almost always, the answer is no. In these examples, the defenders are relying on fundamentally sound defenses, such as a strong cryptosystem or a heavy metal safe. These defenses will not be designed better just because one user's key or one safe's combination is revealed.

In summary, there is a large help-the-attacker" effect from disclosure of a private key, combination to a safe, or password. There is usually no help-the-defender effect. Even for supporters of "no security through obscurity," this sort of information should stay secret.[40]

4. Why Secret Surveillance May Improve Security. The next question is whether it improves security to reveal surveillance techniques used by defenders. Under the Open Source paradigm, one might believe that disclosure will help the defenders because outside experts will suggest how to improve the surveillance system. In addition, the Open Source paradigm would suggest that attackers already know or will readily learn about the defenses of a system, so that disclosure will not help the attackers. The intuitions of intelligence experts are precisely the opposite. These experts believe that it is imperative to keep secret the sources and methods used for intelligence gathering.

The model for uniqueness shows why the latter view is usually better for achieving security. The key factual point is that attackers usually learn little or nothing about surveillance (low ***L***) from their attacks. As the level of ***L*** approaches zero, then attackers do not learn about vulnerabilities even after a high number of attacks. Where ***L*** is so low, the effectiveness of hidden surveillance persists.

---

[39] For the person interested in testing this, simply use a search engine with terms such as "password hacker."

[40] One can imagine a couple of settings where disclosure of the private key may be justified. One reason is if the defender does not deserve to win, such as when the defender is a criminal. Another reason is if a defender won't change a compromised password or private key, even after being told about the vulnerability. Telling that defender that the entire world will learn the password might be the drastic step needed to prompt the change. These examples, however, do not take away from the general point – it almost always helps the attackers more than the defenders to disclose the private key.

To illustrate the difference between surveillance and most physical attacks, return to the example of the machine guns or the hidden pit covered by leaves. The attackers have a high ***L*** from these attacks – they learn about the location of the machine guns or of the existence of the hidden pit. By contrast, suppose that there are well-hidden observers or surveillance cameras that watch the attack. Even attacks that succeed quite possibly would not capture the observer or find out the strategy for the hidden cameras.

The same pattern exists for wiretaps or bugs on networked systems such as telephones or the Internet. A person using the telephone is not supposed to be able to tell if the line is tapped. Even a hacker who gets past a firewall may trigger alarms that the attacker can't perceive. Once again, there is a low ***L*** about surveillance defenses.

Those involved in surveillance have long understood the importance of preventing the opposition from knowing the nature of their surveillance. For instance, Neal Stephenson organizes his masterful novel *Cryptonomicon* around precisely this theme.[41] The novel retells the story of the Allies' decryption of the German Enigma encryption system during World War II. The strategic question for the Allies is how much to act on the secret messages they have decoded. For instance, if a convoy is crossing the Atlantic and the U-boats are poised to attack, should the convoy shift course? If the convoy does, then the Germans might deduce that the Enigma system has been broken, undermining the long-term ability to win the war. If it does not, then many people and boats will be lost. The novel describes elaborate efforts by the Allies to create cover stories for how they get useful intelligence. They seek to reduce the learning (***L***) by the attackers who are subject to surveillance.

The importance of retaining a hidden surveillance capability was also crucial to the entry of the United States into World War I.[42] At a time when Germany and the United States were officially neutral, the Germans sent the famous "Zimmerman telegram" to the government of Mexico. The telegram offered enticements for Mexico to ally with Germany against the United States, including promises of returning to Mexico territories that it held prior to the 1848 war. British intelligence decrypted the communication, but the intelligence agency was extremely loath to reveal to anyone else that it had the capability of breaking German codes. British intelligence then went through an elaborate, and successful, effort to make the leak appear to come from within the Mexican government. The Zimmerman telegram became public, speeding the entry of the United States into the war while retaining the British ability to conduct hidden surveillance of German communications.

These examples highlight the reasons that intelligence experts believe that sources and methods of surveillance should remain secret. They believe that disclosure of sources and methods will increase ***L*** and thus significantly help attackers. Even in many computer security settings, there is often a low ***L*** and surveillance measures can stay hidden. If these factual assertions are correct, as I believe they are, then disclosure of

[41] NEAL STEPHENSON, CRYPTONOMICON (2002).
[42] The account here follows Singh, *supra* note __.

surveillance sources and methods will typically have a large help-the-attacker effect. Persons subject to wiretaps will stop talking on the phone. Persons who know that some radio frequencies are being monitored will shift to other frequencies, and so on.

It is vital to underscore the nature of the claim here. The claim is that hiddenness about surveillance sources and methods will often improve security. This surveillance will improve the ability of defenders to protect their systems from the attackers. The claim is not, however, that hidden surveillance is therefore desirable (or lawful) in any particular setting. The assessment of overall desirability depends on judgments about multiple and often conflicting goals. Wiretaps and other surveillance, for instance, intrude on personal privacy. Fear of surveillance may chill desirable uses of communications networks, with negative effects on the economy and free speech. Public disclosure and debate about surveillance techniques are also crucial to holding government accountable. My current scholarly work on "The System of Foreign Intelligence Surveillance Law" examines these issues of security, privacy, and accountability in great detail.[43] The claim here is about the effectiveness of keeping surveillance hidden. Disclosure about sources and methods will rarely make the surveillance more effective at stopping attacks.

In summary, hidden surveillance techniques mean that there is a low level of learning (*L*) from attacks. Disclosure about the sources and methods of hidden surveillance is likely to reduce security, at least in the short term. Any overall judgment about the desirability of surveillance depends, in addition, on important other values such as the protection of privacy and the accountability of those conducting the surveillance.

5. When Do Attackers Already Know of the Vulnerability? We now can return to the first assumption of the Open Source paradigm: that attackers will learn little or nothing from the disclosure. The discussion here has shown one setting in which attackers learn from the disclosure but in ways that benefit the defenders – attackers will sometimes be deterred when they learn about the defense. The discussion here has also identified two important categories where the assumption seems incorrect. First, private keys and the combinations to safes should stay secret, because the defense is based on the strategy of hiding that information from attackers. Second, surveillance techniques will often not be observable by the attackers, so the assumption that attackers already know about those techniques will often be wrong.

a. Discovering and Exploiting Vulnerabilities. With these important categories better understood, the main debate for software and network security concerns scenarios that do not primarily involve deterrence, private keys, or hidden surveillance. The main scenarios involve the following question – do hackers already know about, or will they promptly know about, the vulnerabilities in a firewall, a mass-market software program, or a cryptosystem?

[43] Peter P. Swire, "The System of Foreign Intelligence Surveillance Law," __ Geo.Wash. L. Rev. __ (forthcoming, 2004).

In answering this question, an important variable is how hard it is for outsiders to discover a vulnerability. If it is generally easy to spot a vulnerability and exploit it, then the Open Source assumption will be correct – additional disclosure will not help the attackers much. Based on my discussions with computer security experts, however, there are at least three reasons to believe that spotting a new vulnerability in a mass-market software program is often more difficult. First, modern software programs often are incredibly complex, involving millions of lines of code.[44] Spotting any individual vulnerability requires considerable searching. There is thus often a lower number of attacks (***N***) on any piece of code than might otherwise be assumed. Second, the greater emphasis on computer security in recent years quite possibly has reduced the number of bugs per line of code. Third, my discussions with professional "bug hunters" suggest that finding a single vulnerability often takes considerable work by a highly skilled person. If one considers a cryptosystem rather than a mass-market software program, this last point is likely to be even more true – it will take a skilled person a considerable amount of work to find a vulnerability if there is one.

If a vulnerability is discovered, the next question is how difficult it is to write the exploit code to take advantage of the vulnerability. If it is hard to write the exploit code, then discovery of the vulnerability will not lead to rapid attacks on the vulnerability. This step may actually be easier for experts to do than one might have suspected. Based on my interviews with computer security experts, for large software programs even an announcement at a high level of generality about a flaw often quickly translates into exploit code. The "progress" from description of the vulnerability to successful attack happens due to high learning about what attacks work (***L***) and high communication with other potential attackers (***C***). Even a fairly general description of a vulnerability can focus skilled attackers on a subset of the millions of lines of code, speeding discovery of the exploit code.

The experts' rapid ability to exploit a vulnerability may or may not translate into non-experts' ability to do the same. At issue is the ability to attack by "script kiddies," the often-young hackers who find a step-by-step script on the Internet for doing an attack. Script kiddies can effectively attack a system that is configured in a standard way and has not installed a patch for a known vulnerability. On the other hand, defenders can often defeat script kiddies by altering the system in some way. In terms of the model developed in this paper, uniqueness (***U***) by the defender aids the defense. Having a unique and hidden defense quite possibly will defeat attackers who are simply following a script.

b. <u>The Analogy Between Exploiting Vulnerabilities and the Efficient Capital Markets Hypothesis</u>. There is a useful analogy to the rich literature on the efficient capital market hypothesis ("ECMH") in economics.[45] Efficiency in the ECMH

---

[44] *See, e.g.*, Dennis Fisher, *Microsoft Puts Meat Behind Security Push*, EWeek, Sept. 30, 2002, *available at* http://www.landfield.com/isn/mail-archive/2002/Oct/0004.html (discussing Microsoft's "massive bug hunt among millions of lines of its Windows code").

[45] Credit for the ECMH is often given to Eugene Fama, "The Behavior of Stock Market Prices," 38 J. Bus. 34 (1965). For a detailed explanation of the ECMH in historical perspective, together with critiques of it,

means that the current price of the stock or other security accurately includes all the relevant information. Efficiency in the Open Source paradigm also means that all the relevant information is already known to outsiders – disclosure of a vulnerability does not help the attackers.

The claim here is that the Open Source paradigm has implicitly assumed what is called the "strong" form of the ECMH, that "current security prices fully reflect all currently existing information, whether publicly available or not."[46] The efficiency of capital markets, in this theory, depends on the actions of a large number of traders who follow the market extremely closely and exploit any opportunity to make a profit. The traders who analyze new information more quickly and accurately than other traders can gain a temporary advantage. The combined effect of all these traders is to push the market very quickly to the point where the price reflects all available information.[47]

The Open Source paradigm makes assumptions similar to the ECMH – attackers learn little or nothing from disclosure, because the attackers have already efficiently figured out the vulnerabilities. One can identify some important differences, however. First, the number of traders in the capital markets is very high, with numerous experts and traders for even the lesser-known stocks. By contrast, there is not necessarily a supply of expert hackers for each aspect of the large computer programs and for each lesser-known computer program. Second, the incentive structure to create "efficiency" is quite different. In the stock market, it is lawful trading that pushes the stock market to efficiency. The successful analyst buys stock and makes money immediately and in large quantities. By contrast, there is seldom a big cash reward for discovering a vulnerability (although the "bug finder" may develop a good reputation and gain consulting contracts). Exploiting the vulnerability also has different incentives – there are criminal penalties for attacking computers, so the incentive to use the knowledge is presumably lower than for lawful stock trading.

These differences would predict that the market for finding computer vulnerabilities is less efficient than the market for finding wrongly-priced securities. The likely inefficiency in finding vulnerabilities undermines the Open Source assumption that attackers already know about vulnerabilities or will promptly discover them. The likely inefficiency is even greater, moreover, in light of the criticisms made against the ECMH itself in recent years. There has been a significant and growing literature showing ways

---

see Lawrence A. Cunningham, "From Random Walks to Chaotic Crashes: The Linear Genealogy of the Efficient Capital Market Hypothesis," 62 Geo. Wash. L. Rev. 546 (1994).

[46] Id. at 560.

[47] The strong version of the ECMH assumes that the market price of the security reflects publicly-available information as well as information known only to the company itself. Under the semi-strong view, insiders might know additional information that would shift the price of the security if publicly revealed. This "insider information" can thus be valuable to insiders because they can predict the price better than public traders. Section 10(b) of the Securities Act of 1934 prohibits insider trading. 15 U.S.C. § 78j(b).

The semi-strong view of the Open Source paradigm, by analogy, would state that insiders might know of vulnerabilities that are unknown to the outside attackers. The efficiency of the market would be determined by how well the outsiders could detect and exploit the vulnerabilities that do not depend on having such insider information.

in which capital markets are not as efficient as the ECMH's proponents had previously thought.[48]

Additional research might fruitfully develop the analogy further between the ECMH and the efficiency of finding vulnerabilities in computer systems. For instance, researchers might explore each of the criticisms of the ECMH in order to examine the possible sources of inefficiency in the exploitation of vulnerabilities. By analyzing the possible sources of inefficiency, computer security researchers can identify areas where vulnerabilities are less likely to be known to attackers, and where disclosure is thus more likely to provide substantial assistance to attackers.

On the other hand, there are scenarios where attackers have very strong incentives to discover vulnerabilities. In future military conflicts, for instance, attackers will be highly motivated to discover any vulnerability in the computer or network systems held by their enemy. Where there are determined and well-financed adversaries, the attackers may be very effective in discovering vulnerabilities. One can therefore imagine the following counter-intuitive situation. Civilian attackers may be inefficient, so that disclosure has a large help-the-attacker effect. Military attackers, by contrast, may be efficient in exploiting vulnerabilities that can be perceived from the outside. For those vulnerabilities, greater disclosure might actually be rational. The disclosure will do little or nothing to help the attackers, but there may be help-the-defender effects for system designers or for other defenders who rely on the system.

In summary, there is likely greater inefficiency today in the discovery of computer and network vulnerabilities than assumed in the Open Source paradigm. The analogy to the Efficient Capital Markets Hypothesis shows that the degree of efficiency depends on the incentives and institutional arrangements that attackers have to discover and communicate about vulnerabilities.

[48] *See, e.g.,* William T. Allen, "Securities Markets as Social Products: The Pretty Efficient Capital Market Hypothesis," 28 J. Corp. L. 551 (2003); Lynn A. Stout, "The Mechanisms of Market Inefficiency: An Introduction to the New Finance," 28 J. Corp. L. 635 (2003).

B. The Assumption that Disclosure Will Tend to Improve the Design of Defenses.

The next assumption is the one most strongly held by Open Source proponents -- disclosure of code and vulnerabilities will improve security because it will result in improved design of defenses. As firewall experts Chapman and Zwicky have written: "Some people feel uncomfortable using software that's freely available on the Internet, particularly for security-critical applications. We feel that the advantages outweigh the disadvantages. You may not have the 'guarantees' offered by vendors, but you have the ability to inspect the source code and to share information with the large community that helps to maintain the software. In practice, vendors come and go, but the community endures."[49]

In this paper, I do not take a position on the almost theological issue of whether Open Source software provides better security than proprietary software.[50] Instead, the discussion here will seek to identify some of the variables that would tilt the outcome in one direction or the other. The previous discussion showed how mass-market software and firewalls are subject to more efficient attacks due to the high number of attacks (***N***), learning from attacks (***L***), and communication among attackers (***C***). The focus here is on how defenders can alter the defenses (***A***) in ways that improve the defense over time. After looking at variables that affect when Open Source or proprietary software may provide better security, the discussion turns to how openness has particular value in promoting security and accountability in the long run.

1. Variables that Affect When Open Source or Proprietary Software May Provide Better Security. Consistent with the goals of this paper, the effort here is to identify situations where openness is more or less likely to improve security. In this discussion, it is helpful to distinguish between two meanings of "open." The focus of the discussion in this paper is on "open" in the sense of "not hidden." In particular, outsiders can generally see the source code for Open Source software but not for proprietary software. This paper does not address the extent to which software should be "open" in the sense of "not owned" under copyright or other laws.

a. Expertise of Inside and Outside Programmers. The security of proprietary software relies substantially on the expertise of "inside" programmers. These individuals are employees or contractors of the company that owns the mass-market software or the organization that operates the firewall.[51] By contrast, the Open Source paradigm relies on "outside" programmers – individuals who usually are not employed by or under contract with whomever initially designed the software.

---

[49] D. Brent Chapman & Elizabeth D. Zwicky, Building Internet Firewalls 23 (1995).

[50] In the interests of full disclosure, I note that I am a member of Microsoft's Trustworthy Computing Academic Advisory Committee, which is a group of 19 academics that has been asked to provide advice on security and privacy issues to Microsoft. I have also discussed the issues in this paper at great length with many Open Source advocates. The views expressed herein are entirely my own.

[51] Proprietary organizations may also get tips about problems and solutions from users of the software and other outsiders, but the emphasis is likely to be on inside programmers.

The respective effectiveness of either the Open Source or the proprietary approaches will depend on the relative quantity and expertise of inside and outside programmers. Chapman and Zwicky emphasized the advantage of outside programmers when they referred to "the large community that maintains the software."[52] In other settings, an organization might be able to bring more and better programmers, who have the relevant expertise, to the inside. For example, consider the software for specialized military uses such as for launching rockets. In such a setting, there may not be a "large community [on the outside] that maintains the software."[53] The more effective approach, in the absence of that outside community, quite likely would be to rely on inside programmers - persons who are hired or trained by the military. In such instances, disclosure of the software does not enlist a community of outside programmers, although it may help the attackers find vulnerabilities.

b. The Incentives to Improve the Defense. One of the chief arguments for supporters of the proprietary approach is that the owner of the software or the system has strong incentives to provide security. The reputation of the software manufacturer or system owner is on the line, and bad security can lead to a direct loss of revenue. In the Open Source model, the incentives are less clearly defined. The outside programmers might gain a good reputation by designing a patch. A good reputation might translate into consulting contracts or other remunerative work, although the time spent working on a patch seems less directly profitable for the Open Source programmer than it is for a company that increases sales due to better security. Open Source programmers may also improve software due to a combination of other motives, including membership in a community of programmers and the feeling of satisfaction from helping solve the problems facing others.

The extent to which one approach - proprietary or Open Source - will provide greater incentives to improve the defense is essentially a question of sociology and organizational behavior. Over time the sociological context might shift. A vibrant Open Source community in one period might descend into a "what's in it for me?" approach in a later period. Alternatively, a vibrant Open Source community might become broader and deeper in its skills over time compared with the inside programmers available to proprietary efforts.

c. Persistence of the Expertise. Chapman and Zwicky point out the risk that any single company can disappear: "In practice, vendors come and go, but the community endures." Users of a proprietary product thus risk the chance that the company will go bankrupt or otherwise stop supporting the product. On the other hand, there are scenarios where the proprietary approach would likely lead to better persistence of expertise. The owner of a software program or a firewall might invest in having specialized expertise.[54] For instance, the owner of a software program may find it

[52] Chapman & Zwicky, *supra* note __, at 23.
[53] Id.
[54] The economist Oliver Williamson calls this sort of investment "transaction specific capital" and considers it an important predictor of where firms make investments. Oliver E. Williamson, The Economic Institutions of Capitalism 30-32 (1998).

worthwhile to keep on staff a person who is expert in one complex piece of a software program. Similarly, the military might decide to keep on staff persons who are experts in software that only the military uses. In these instances, the proprietary model may well create more persistent expertise than an Open Source approach.

d. The Institutional Context for Patching. The usual Open Source belief has been that patching – the release of improved code that addresses a vulnerability – works better where the Open Source community can probe for vulnerabilities and then fix them. The accompanying belief has been that many proprietary companies have been reluctant to admit to vulnerabilities or to invest the resources to issue good patches.

My interviews with computer security experts suggest that these conclusions have quite possibly become less true over time. First, proprietary companies have shifted to a norm of issuing patches as part of the overall effort to improve cybersecurity. Second, proprietary companies have in some instances created substantial institutional structures to create and disseminate patches. These institutional structures can respond to vulnerabilities in a coordinated way. A coordinated approach, if carried out effectively, may lead to faster and more consistent responses to problems across different platforms.

The point here is not to announce that one approach or the other is necessarily the winner when it comes to effective patching. Instead, the speed and quality of patching is likely to vary over time depending on the institutions that exist to create and disseminate patches.

e. Interoperability and Openness. Having Open Source code can facilitate interoperability. System owners will have the ability to see what they are including in their system and how to integrate it with existing programs. For this reason, greater disclosure can improve security to the extent it permits system owners to know their systems and avoid vulnerabilities.

On the proprietary side, there is the usual market incentive to provide effective solutions for clients. Software companies want their products to work well for a large range of users. They often design their products to inter-operate with other products, and they work with system integrators to create overall systems that work.[55] As with other factors discussed in this section, the extent to which the possible advantages of openness outweigh the possible advantages of vendors directly seeking to satisfy the market by increasing sales is an empirical question..

2. The Role of Disclosure in Creating Long-Run Security and Assuring Accountability. Much of the comparison thus far of the Open Source and proprietary approaches implicitly concerned short and medium-term security, such as which approach would typically create a better patch for a newly discovered vulnerability. An additional basis for disclosing information is to improve *long-run* security. Bruce

[55] For a discussion of the incentives for software and other manufacturers to promote interoperability, see Joseph Farrell & Philip Weiser, *Modularity, Vertical Integration, and Open Access Policies: Towards a Convergence of Antitrust and Regulation in the Internet Age*, 17 HARV. J. L. & TECH. 85, 97-104 (2003).

Schneier, for instance, writes that "public scrutiny is the only reliable way to improve security -- be it of the nation's roads, bridges and ports or of our critically important computer networks."[56] The belief is that organizations that rely on secrets are unlikely in the long-run to update their security effectively. On this view, testing by outsiders is crucial to overcoming inertia within the organization. Even if secrecy masks vulnerabilities in the short-run, the secretive organization is laying the groundwork for a larger-scale problem in the long-run.

It is difficult to provide empirical tests for when secrecy leads to long-run failure to adapt and modernize. One can try to compensate for secrecy by creating institutions that inspect and challenge the status quo without disclosing secrets to the world. Organizations can hire "Tiger Teams"[57] and other sorts of outside experts to probe for weaknesses. Organizations can hire independent auditors and create oversight boards to watch for areas of weakness. The usefulness of these institutional responses will vary widely, but they are unlikely to be effective unless they can probe into possible areas of vulnerability and then have institutional support when they recommend changes.

Over the long run, the usefulness of openness for promoting security overlaps with the usefulness of openness for assuring accountability more generally. The Freedom of Information Act[58] and other openness mechanisms are useful in part because they allow vulnerabilities to be discovered and security to be improved. These mechanisms are also useful, however, for exposing corruption, abuse of power, and the other evils that can flourish in secret. It is a topic of my continuing research to shed light on situations where openness is most important to accountability and long-run improvement in security. For purposes of this paper, however, the claim is more modest. Once one has done the analysis on the extent to which disclosure helps security, there is reason to place a thumb (and perhaps an entire palm) on the scale on the side of disclosure. That tilt is due to the recognition of the likely long-run decrease in security and accountability that comes from secrecy. The longer that information is designed to stay secret, the greater the risk to system security and general accountability.

On the comparison of Open Source and proprietary software, this paper does not take a position on the contentious issue of which approach provides better overall security. Significant variables include: the relevant expertise of inside and outside programmers; the incentives to improve the defense; the persistence of relevant expertise; the institutional context for patching; and how interoperability is assured. Disclosure is often additionally useful for promoting long-run security and assuring accountability.

---

[56] Bruce Schneier, "Internet Shield: Secrecy and Security," S.F. Chronicle, Mar. 2, 2003.
[57] Michael Lee et al., *Electronic Commerce, Hackers, and the Search for Legitimacy: A Regulatory Proposal*, 14 Berkeley Tech. L.J. 839, 884 n. 195 (1999) (defining "Tiger team" as computer security experts, hired by the owner of a computer system, who simulate hostile break-ins).
[58] 5 U.S.C. § 552.

C. The Assumption that Disclosure Will Spread Effective Defenses to Others.

The third assumption in the Open Source paradigm is that disclosure will spread effective defenses to third parties. The assumption in the military paradigm is that disclosure will prompt little or no improvement in defense by other defenders. The discussion here will seek to explain when each assumption is more likely to be true and thus when disclosure is likely to help other defenders.

The military assumption is more convincing in settings where mechanisms exist to disclose only to trusted defenders. In the military setting, there are strongly-authenticated fellow defenders, such as the others that share the same uniform. When changes should be made in the defenses, the military has a hierarchical command structure. Orders can be given to the appropriate units to implement the change in defense. Under these assumptions of strong authentication and an established hierarchy, disclosure to the entire world has low or zero benefits (the other defenders improve their defenses) and potentially significant costs (the help-the-attacker effect).[59]

The situation changes for mass-market software. There is no strong authentication for who is an "authorized security specialist" for widely-used software. Suppose that a large software company tried to send security information to every "authorized security specialist" while trying to keep the information secret from all potential hackers. That sort of mass notification, with no leakage, is highly unlikely to succeed. In the absence of strong authentication that separates "good guys" from "bad guys," the disclosure that does occur will generally be available to both. The military option of selective disclosure is much less likely to be available.

The mass-market software programmer also has less hierarchical control over defenses than does a military commander. For mass-market software, many users lack expertise. Many defenders also may not pay much attention to the programmer's plea to install patches or otherwise upgrade the defense. Given the lack of hierarchical control, those seeking to spread the new defensive measure may have to rely on widespread publicity to alert the third-party defenders about the threat.

In short, disclosure is more likely to help attackers where there is a unified defense (one organization with hierarchical controls). Disclosure is more likely to help defenders where there are numerous third parties that risk harm from a vulnerability. The latter situation is more likely to occur for the major settings for computer and network security. For firewalls and mass-market software there are many ordinary users who might have the vulnerability. For encryption, the messages of many users are subject to attack if the cryptosystem is broken. Because there are so many third parties, disclosure

[59] In the real life of the military, of course, the assumptions of strong authentication and effective hierarchy do not always exist. Spies might learn about information that was supposed to go only to members of the military. Orders might not be followed. Nonetheless, the ability to get information to selected fellow defenders is likely much greater in a well-run military organization than it is for mass-market software companies.

becomes more important in order to alert defenders about whether a product is secure or whether a patch is needed.

Interestingly, the needs of the U.S. military seem to have played a role in prompting mass-market software companies to disclose more about vulnerabilities. Over time, the military has followed the dictates of Congress and bought a great deal of COTS (commercial off-the-shelf) software. Based on my interviews with security experts, [60] the military became aware over time of software vulnerabilities that had not been disclosed either to the military or to the general public. The military found it unacceptable to be vulnerable to attacks that were known to attackers and to the software company, but not to the military. The military thus put pressure on software providers to increase the disclosure of vulnerabilities, so that users such as the military would be in a better position to know about vulnerabilities and develop a response.

In summary, disclosure will tend to help the attackers but not the defenders in a military setting, where there is strong authentication of defenders and an established hierarchy to implement better defenses. Disclosure provides greater benefit to defenders when there are numerous third-party users, no effective way to communicate only to friendly defenders, and no hierarchical way to ensure that defenses are put into place.

IV. Conclusion: Security, Privacy, and Accountability.

This paper has addressed when disclosure of information will improve security. "Security," for this paper's purposes, is defined as preventing the attacker from gaining control of a physical installation or computer system.[61]

There are clearly other compelling goals to consider in deciding when to disclose information. Accountability usually increases with greater disclosure. Disclosure of information can produce economic winners and losers. Free speech and other First Amendment issues are implicated by disclosure policy. Personal privacy, the subject of much of my previous academic and government work, can also be compromised when

---

[60] The information about this history was provided to me on background by persons in both the public and private sector.

[61] A more expansive definition of "information security" is given in the Federal Information Security Management Act of 2002, Pub. L. 107-347, 116 Stat. 2946, Sec. 301:

> "(1) The term 'information security' means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide--
>
> "(A) integrity, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity;
>
> "(B) confidentiality, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information; and
>
> "(C) availability, which means ensuring timely and reliable access to and use of information.

information is disclosed[62] Compelling goals such as accountability, economic growth, free speech, and privacy should be included in any overall decision about whether to disclose information.

An essential part of the analysis, however, is to understand when disclosure helps security itself. Understanding this is important in its own right, as an intellectual topic that not that has not received sufficient attention to date. It is also crucial in the debates about how to create cyber-security and physical security in the age of the Internet and of terrorist threats.

This paper seeks to correct two common misunderstandings about when disclosure improves security. Secrecy helps much more often than is suggested by the slogan that "there is no security through obscurity." In presenting earlier versions of this paper, the most sophisticated technologists have understood this fact. They have known that keys and passwords should remain secret and that a good firewall can benefit from idiosyncratic features that defeat the script kiddies. This paper draws on the literature about Efficient Capital Markets Hypothesis, however, to suggest that the efficiency of attackers in discovering vulnerabilities will often be less than Open Source proponents have presumed. More broadly, my discussions with security experts have uncovered no models or systematic ways to analyze the limits of what I have called the Open Source paradigm for security through openness.

The paper also teaches that disclosure improves security much more often than is suggested by the slogan that "loose lips sink ships." First, military systems often rely on commercially-available software, and the security of those systems thus depends on military system owners learning about vulnerabilities. Second, military actions are subject to the growth of the Public Domain, where information gets communicated so quickly and effectively among potential attackers.[63] Third and most broadly, the model in this paper suggests that openness may be the best general strategy in situations with

---

[62] From 1999 until early 2001 I served as the Clinton Administration's Chief Counselor for Privacy, in the U.S. Office of Management and Budget This research project arises from my perception at that time that some of the hardest and least understood issues concerned how privacy and security should intersect. My work on the topic began in the summer of 1999, when the Federal Intrusion Detection Network (FIDNet) became a topic of controversy. In the wake of criticism of FIDNet (*see* John Markoff, "U.S. Drawing Plan That Will Monitor Computer Systems," N.Y. Times, July 28, 1999, at A1), I was asked to work with Richard Clarke's cyber-security team to ensure that federal computer security was done consistently with privacy and civil liberties. The next year, I served as chair of a White House Working Group on how to update electronic surveillance laws for the Internet Age, another topic where privacy and security concerns intersected. Since my return to academic life, many of my writing have addressed the intersection of security, privacy, and surveillance issues. My privacy and other publications are available at www.peterswire.net.

[63] The growth of the Internet, with its lack of national boundaries on communications, has lowered the cost and increased the effectiveness of research about other countries. Military commanders expect to use new technologies to "see the battlespace" and have "integrated sight" of the enemy's capabilities. ADMIRAL BILL OWENS, LIFTING THE FOG OF WAR 119, 133 (2000) (describing greater information gathering and processing as a central part of the "Revolution in Military Affairs"). Opposing forces will similarly pursue strategies of high ***N***, ***L***, and ***C*** to "life the fog of war."

low uniqueness, where there are high values for number of attacks (***N***), learning from an individual attack (***L***), and communication among attackers (***C***).

In terms of methodology, this paper has offered an economic analysis for determining when "there is no security through obscurity" and when "loose lips sink ships." The first step is to assess the costs and benefits of the disclosure with respect to potential attackers. In some instances, for strong positions, disclosure will deter attacks and is thus beneficial. In other instances, disclosure tends to spread information about vulnerabilities. Even then, where the facts fit the Open Source and Public Domain paradigms, disclosure will offer little or no aid to attackers. Thus, disclosure can go forward if there are benefits to the defenders or if other values favor disclosure. When the facts fit the Information Sharing and Military paradigms, disclosure is more likely to help the attackers. Nonetheless, disclosure is more likely than previously to have net benefits in cyber-settings and other modern settings where attackers can mount numerous attacks (high ***N***), gather information cheaply (high ***L***) and communicate efficiently about vulnerabilities (high ***C***).

Another important theme of the paper is that there are often third-party defenders who benefit from disclosure about vulnerabilities. The Military Paradigm implicitly assumes that defenders act in a unified way with strong authentication (the ability to recognize allied soldiers) and hierarchical control (the ability to order fixes for vulnerabilities). When these assumptions no longer hold, such as for mass-market software and networks operated by diverse actors, then disclosure is much more likely to have net benefits for defenders.

By defining the factors that contribute to high uniqueness, the paper identifies the variables that determine when secrecy will improve the defense: the effectiveness of the defensive feature against the initial attack (***E***); the number of attacks (***N***); the degree of learning by the attacker (***L***); the degree of communication with other potential attackers (***C***); and the extent to which defenders can effectively alter the feature before the next attack (both alteration by the system designer (***A-D***) and alteration by third parties, such as Open Source programmers (***A-T***)).

Identification of these variables provides the answer to the question asked in the paper's title: What is different about computer and network security? For key computer and network topics such as firewalls, mass-market software, and encryption, the effect of variables such as high ***N***, ***L***, ***C***, and ***A-T*** show why the benefits of disclosure of vulnerabilities often outweigh the benefits of secrecy. Disclosure is not necessarily or logically more desirable for computer and network security than for physical security, but the crucial variables much more often result in having net benefits from disclosure. Examination of the variables also illuminates important special cases, such as why disclosure of passwords and private keys will almost always be harmful and why surveillance generally benefits from secrecy concerning sources and methods.

In closing, the intellectual structure of this paper provides a systematic way to identify the costs and benefits of disclosure for security. Further research can assess the

empirical levels of the relevant variables in different security contexts. Further research can enrich the theoretical structure for assessing the effects of disclosure on security, such as by drawing more on the Efficient Capital Markets Hypothesis literature to identify where vulnerabilities are most likely to be discovered by attackers. Finally, further research can better explain how security goals should be integrated with other compelling goals such as accountability, economic growth, free speech, and privacy.