VULNERABLE SOFTWARE: PRODUCT-RISK NORMS AND THE PROBLEM OF UNAUTHORIZED ACCESS*

Richard Warner†
Robert H. Sloan††

Abstract

Unauthorized access to online information costs billions of dollars per year. Software vulnerabilities are a key cause of these losses. Software currently contains an unacceptable number of vulnerabilities. The standard solution notes that the typical software business strategy is to keep costs down and be the first to market, even if that means the software has significant vulnerabilities. Many endorse the following remedy: make software developers liable for negligent or defective designs. This remedy is unworkable. We offer an alternative based on an appeal to product-risk norms. Product-risk norms are social norms that govern the sale of products. A key feature of such norms is that they ensure that the design and manufacture of products impose only acceptable risks on buyers. Unfortunately, mass-market software sales are not governed by appropriate product-risk norms; as a result, market conditions exist in which sellers profit by offering vulnerability-ridden software. This analysis entails a solution: ensure that appropriate norms exist. We contend that the best way to do so is a statute based on best practices for software development, and we define the conditions under which the statute would give rise to the desired norm. Why worry about creating the norm? Why not just legally require that software developers conform to best practices? The answer is that enforcement of legal requirements can be difficult, costly, and uncertain; once the norm is in place, however, buyers and software developers will conform to best practices on their own initiative.

* This Article is based upon work supported by the National Science Foundation under Grant No. IIS-0959116.

† Professor of Law, Chicago-Kent College of Law; Visiting Foreign Professor, Law Faculty, University of Gdańsk, Poland.

†† Professor and Head, Department of Computer Science, University of Illinois at Chicago.

TABLE OF CONTENTS

Losses from unauthorized access to online information run in the billions per year.[1] We assume it would be better to avoid these losses;[2] our question is how best to do so. We limit our inquiry by focusing exclusively on one significant source of unauthorized access: software vulnerabilities.[3] A vulnerability is a property of a software system that could be exploited to gain unauthorized access to a computer or network.[4] The prevailing and correct consensus is that software programs currently contain an unacceptable number of vulnerabilities.[5] Why? And what is the remedy? The standard answer to the first question assumes that "businesses are profit-making ventures, so they make decisions based on both short- and long-term profitability."[6] Reducing vulnerabilities requires a longer and more costly development process, and the typical profit-maximizing strategy is to keep costs down and be the first to

1. In 2009, the cost of a data breach to organizations in the United States was an average $6.75 million per incident. PONEMON INST., 2009 ANNUAL STUDY: COST OF A DATA BREACH 3 (Jan. 2010), *available at* http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file/US_Ponemon_CODB_09_012209_sec.pdf. *See also* Robert W. Hahn and Anne Layne Farrar, *The Law and Economics of Software Security*, 30 HARV. J.L. & PUB. POL'Y 283, 302–08 (2007) (examining the calculated cost of a security breach). A United Kingdom government study estimates the yearly cost of data breaches to be £21bn to businesses, £2.2bn to government and £3.1bn to citizens. DETICA, THE COST OF CYBERCRIME 2 (Feb. 2011), *available at* http://www.cabinetoffice.gov.uk/sites/default/files/resources/the-cost-of-cyber-crime-full-report.pdf. Earlier U.S. estimates of the cost of identity theft alone are also in the billions. For a summary of relevant studies, see FRED H. CATE, CTR. FOR INFO. POL'Y LEADERSHIP AT HUNTON & WILLIAMS, INFORMATION SECURITY BREACHES AND THE THREAT TO CONSUMERS 6 (Sept. 2005), *available at* http://www.fredhcate.com/Publications/Information_Security_Breaches.pdf (reporting 10.1 million victims of identity theft in 2003 and a total losses to consumers of over 50 billion). The number of victims has declined recently but costs have actually risen. Jennifer Saranow Schultz, *The Rising Cost of Identity Theft for Consumers*, N.Y. TIMES (Feb. 9, 2011, 12:01 PM), http://bucks.blogs.nytimes.com/2011/02/09/the-rising-cost-of-identity-theft-for-consumers/?src=busln (noting that "[t]he average consumer out-of-pocket cost due to identity fraud increased to $631 per incident in 2010, up 63% from $387 in 2009. Such costs include the expenses of paying off fraudulent debt as well as resolution fees, such as legal costs"). Schultz summarizes this report: JAVELIN STRATEGY & RESEARCH, 2011 IDENTITY FRAUD SURVEY REPORT: IDENTITY FRAUD DECREASES – BUT REMAINING FRAUDS COST CONSUMERS MORE TIME & MONEY, (Feb. 2011), *available at* https://www.javelinstrategy.com/uploads/1103.R_2011%20Identity%20Fraud%20Survey%20Report%20Brochure.pdf (reporting the increasing cost to consumers of identity theft).

2. We assume that the gains from allowing unauthorized access (e. g., saving the time, effort, and money otherwise spent in prevention) are not sufficient to offset the losses. *See infra* note 98 and accompanying text.

3. Vulnerabilities are a major cause of unauthorized access. In 2010, CWE (Common Weakness Enumeration) and SANS (SysAdmin, Audit, Network, Security) identified cross-site scripting (XSS), SQL injection, and buffer overflow vulnerabilities as the causes of nearly all major cyber attacks in recent years. *CWE/SANS TOP 25 Most Dangerous Software Errors*, SANS INST., http://www.sans.org/top25-software-errors (last updated June 27, 2011). When releasing the list, the SANS noted that "[t]hese 25 programming errors, and their 'on the cusp cousins' have been the cause of nearly every major type of cyber attack, including recent penetrations of Google, power systems, military systems, and millions of other attacks on small businesses and home users." Joan Goodchild, *Security Experts: Developers Responsible for Programming Problems*, CSO (Feb. 16, 2010), http://www.csoonline.com/article/544163/Security_Experts_Developers_Responsible_for_Programming_Problems. (citation omitted). *See also Applications Security: Eliminating Vulnerabilities in Enterprise Software*, INFO. WEEK, 1 (July 9, 2010), http://i.cmpnet.com/darkreading/vulnerabilitymgmt/July2010_ApplicationsSecurity.Alert[1].pdf (noting that "[m]ost of the hacks that compromise enterprise security today are those that exploit flaws in applications").

4. *See, e.g.*, ROSS ANDERSON, SECURITY ENGINEERING 15 (Wiley Publ'g Inc., 2d ed. 2008) (defining a vulnerability as "a property of a system or its environment which, in conjunction with an internal or external threat, can lead to a security failure, which is a breach of the system's security policy").

5. *See, e.g.*, Bruce Schneier, *Information Security and Externalities*, BRUCE SCHNEIER (Jan. 2007), http://www.schneier.com/essay-150.html (discussing information security as an economic problem).

6. *Id.*

offer a particular type of software, even if it is imperfect in a variety of ways, including having vulnerabilities.[7] Many who offer this diagnosis endorse the following remedy: make software developers liable for negligent or defective designs—either by adapting common law tort doctrines or by enacting statutes based on negligence or product liability concepts.[8] We do not dispute the profit-maximizing diagnosis. "The market often rewards first-to-sell and lowest cost rather than extra time and cost in development."[9] But we do reject the remedy. We offer an alternative based on an appeal to product-risk norms.

Product-risk norms are social norms that govern the sale of products. A key feature of such norms is that they ensure the design and manufacture of products impose only acceptable risks on buyers.[10] Unfortunately, mass-market software sales are not governed by appropriate product-risk norms; as a result, market conditions exist in which sellers can, and do, profit by offering vulnerability-ridden software. This analysis entails a solution: ensure that appropriate norms exist. We contend that the best way to do so is a statute based on best practices for software development. Our concern with the norm may seem puzzling. Since we will suggest a statute, why not just stop there? Why not just legally require that software developers conform to best practices and not worry about creating a norm? Our answer is that there are significant advantages to creating the norm. Enforcement of the legal requirement can be difficult, costly, and uncertain;[11] once the norm is in place, however, buyers and software developers will conform to best practices on their own initiative.

---

7. *See generally* C. SHAPIRO & H. R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY 50–51 (1999). The economics and information security community has developed Shapiro and Varian's initial insights. Much of this work has been reported in the annual Workshop on the Economics of Information Security since 2002. For information the workshops from 2002 to 2010, see http://weis2010.econinfosec.org/index.html. For a good general survey, see Ross Anderson & Tyler Moore, *Information Security: Where Computer Science, Economics and Psychology Meet*, 367 PHIL. TRANSACTIONS ROYAL SOC'Y A 2717, 2721–22 (2009).

8. Bruce Schneier is a prominent advocate of this view. *See* Bruce Schneier, *Liability changes everything*, BRUCE SCHNEIER, (Nov. 2003), http://www.schneier.com/essay-025.html (arguing that "[i]f we expect software vendors to . . . invest in secure software development processes, they must be liable for security vulnerabilities in their products"). The theme appears frequently in the law review literature. *See, e.g.*, Jennifer A. Chandler, *Improving Software Security: A Discussion of Liability for Unreasonably Insecure Software*, *in* SECURING PRIVACY IN THE INTERNET AGE 155, 166 (Anapum Chander et al. eds., 2006); Shuba Gosh & Vikram Mangalmurti, *Curing Cybersecurity Breaches Through Strict Products Liability*, *in* SECURING PRIVACY IN THE INTERNET AGE 187, 195; STEWART D. PERSONICK & CYNTHIA A. PATTERSON (eds.), CRITICAL INFORMATION INFRASTRUCTURE PROTECTION AND THE LAW: AN OVERVIEW OF KEY ISSUES 50 (2003); David Gripman, *The Doors Are Locked but the Thieves and Vandals Are Still Getting In: A Proposal in Tort to Alleviate Corporate America's Cyber-Crime Problem*, 16 J. MARSHALL J. COMPUTER & INFO. L. 167, 175 (1997); Michael L. Rustad & Thomas H. Koenig, *The Tort Of Negligent Enablement Of Cybercrime*, 20 BERKELEY TECH. L.J. 1553, 1586 (2005) (arguing for recognizing a negligent enablement tort to provide an incentive to avoid negligent design practices); Michael D. Scott, *Tort Liability for Vendors of Insecure Software: Has the Time Finally Come?* 67 MD. L. REV. 425, 467 (2008).

9. Eugene H. Spafford, *Remembrances of Things Pest*, 53 COMM. ACM 35, 36 (2010).

10. *See infra* Part III.

11. *See generally* Paul S. Atkins & Bradley J. Bondi, *Evaluating the Mission: A Critical Review of the History and Evolution of the SEC Enforcement Program*, 13 FORDHAM J. OF CORP. & FIN. L. 367, 414 (2008) (analyzing the enforcement program of the Securities and Exchange Commission); Robert L. Glicksman & Dietrich H. Earnhart, *The Comparative Effectiveness of Government Interventions on Environmental Performance in the Chemical Industry*, 26 STAN. ENVTL. L.J. 317, 367 (2007) (analyzing the chemical industry's compliance with federal environmental regulations).

In Part I, we define norms generally and explain the special case of coordination norms. We also introduce the concept of a value-optimal norm. This concept is required for our central claim: that the sale of mass-produced software is not governed by value-optimal norms. In Part II, we provide the background essential to defending this claim. We argue that product-risk norms are coordination norms that ensure that most buyers demand similar features in particular types of products. We offer three examples. These examples illustrate that in mass markets, product-risk norms are coordination norms that promote buyers' interests by unifying their demands. A mass-market buyer cannot unilaterally ensure that sellers will conform to his or her requirements; instead, coordination norms create collective demands. In Part III, we adapt a well-known law and economics argument to explain why—under ideal transaction conditions—sellers conform to product-risk norms because offering norm-conforming products is the profit-maximizing strategy. As we explain in detail in Part III, transaction conditions are ideal when two conditions hold: there are enough value-optimal product-risk norms, and the market is perfectly competitive. We argue in Part IV that software sales fail to adequately approximate this ideal. While we address the concern that sufficiently competitive markets may not exist, we focus primarily on the lack of appropriate value-optimal norms. We argue that buyers are trapped in a product-risk norm that is not value-optimal, and we contend that the solution is to replace the current norm with a norm formulated in terms of best practices for software development. Part V defines the notion of a best practice, and argues that best practices exist for software development. Part VI specifies the conditions under which a best-practices norm will arise. In Part VII, we argue that legal regulation is required to fulfill these conditions, and we sketch an appropriate statute.

I. NORMS

We begin by describing the purchase of a typical consumer good. When Alice discovers that her water heater no longer works, she purchases a new one. She takes it for granted that the gas pilot light will not stop burning every few days; that the water heater will not burst; that the materials are sufficiently corrosion resistant; that the water heater will function properly for about ten years; and so on. Alice does not try to confirm these assumptions. She does not investigate the water heater, its design specifications, or its manufacturing process. She simply assumes that its design and manufacture do not impose unacceptable risks (as long as she uses the water heater for its intended purpose). She assumes this because she assumes that the sale of the water heater is governed by relevant product-risk norms. This raises three questions. What are the relevant norms? Why, and in what sense, do norm-compliant sales ensure only acceptable risks? And why do buyers and sellers comply with product-risk norms? Clarifying the relevant notion of a norm is an essential preliminary task.

Product-risk norms are coordination norms. Coordination norms are one important species in the broad genus of norms in general. We define the genus

first. A norm is a behavioral regularity in a group where the regularity exists at least in part because almost everyone thinks that he or she ought to conform to the regularity.[12] Suppose, for example, that the norm in Jones's small town is to go to a Protestant church on Sunday; that is, almost everyone goes to a Protestant church on Sunday (even though there is a Catholic church nearby in the next town), and almost everyone does so at least in part because almost everyone believes he or she ought to go to a Protestant church on Sunday. As the "almost" in "almost everyone" indicates, the existence of a norm does not require universal compliance. We will not consider the interesting question of how many count as "almost everyone," and we will, for convenience, drop the "almost" and simply understand "everyone" as "almost everyone."

Norms evolve over time through repeated patterns of interaction; the interactions may initially have their source in custom, private agreement, or law (or a combination of these factors).[13] To take custom first, it is easy to imagine it as the source of the "Protestant church" norm. Suppose it was customary for some part of the town's population to attend the church; church-goers and non-church-goers alike notice the custom, and both groups begin to think that they ought to conform—either out of religious conviction, or some other view of why it is a good thing, or in order to avoid the disapproval of others. Attendees continue to attend while non-attendees increasingly become attendees. To illustrate private agreement, imagine that two years ago Scott and Zoe agreed to meet at Starbucks every morning; having done so for two years, each thinks he or she ought to meet the other at Starbucks. The norm of driving on the right illustrates the role of legal regulation. The norm owes its existence at least in part to the fact that it is the law that one drives on the right. Legal regulation does not, however, always bring a norm into existence; it is the law that one should obey speed limits, but the norm is to exceed them. We defer further consideration of the generation of norms to Parts VI and VII. Until then, we focus on why people continue to conform to already established norms.

A. Coordination Norms

We first define coordination norms and then turn to the question—critical for our later purposes—of why people conform to coordination norms.

1. Coordination Norms Defined

Driving on the right is an example of a coordination norm. Before considering what makes this a *coordination* norm, note that the general definition of the genus is fulfilled: everyone (in the United States) drives on the right, and they do so in part because they think they ought to. Exceptional

12. *See* Michael Hechter & Karl-Dieter Opp, *What Have We Learned About the Emergence of Social Norms?*, *in* SOCIAL NORMS 394, 403 (Michael Hechter & Karl-Dieter Opp eds., 2001) (exploring the varied definitions of the term "norm").

13. *See* DAVID K. LEWIS, CONVENTION: A PHILOSOPHICAL STUDY 5–42 (1969).

circumstances aside, no one thinks he or she should drive on the left—as long as everyone else drives on the right. The "as long as" is the distinctive feature of the example. The "ought" is conditional. Everyone thinks he or she ought to drive on the right, but only on the condition that everyone else does so. If everyone started driving on the left, no one would think he or she ought to drive on the right. This conditional "ought" distinguishes driving on right from the norms we examined earlier. In the Protestant church norm, for example, each churchgoer expects others to attend, but attendance does not depend on that expectation; each attends because each thinks he or she ought to, no matter what others do. "Attend a Protestant church" and "drive on the right" are both norms, but they are different species of the same genus; the latter is a coordination norm, but the former is not. A coordination norm is a behavioral regularity in a group, where the regularity exists at least in part because almost everyone thinks that he or she ought to conform to the regularity, as long as everyone else does.[14] The "ought" is conditioned on the assumption about everyone else. We will need to refer to such "oughts" frequently, and to avoid constant repetitions of "as long as everyone else does," we will often say that, for short, that one thinks one ought *conditionally* to conform.

An example is helpful. You are about to enter an elevator in which others are already present. Where do you stand? The norm is to maximize the distance between you and the person nearest you.[15] Thus, everyone thinks he or she ought to conform to the norm—but *conditionally*, as long as everyone else conforms. There is little point in being the only "nearest neighbor distance maximizer" if everyone else is just going to stand wherever they like. The example illustrates an important feature of coordination norms: they make it possible for parties to coordinate their behavior in ways that realize shared interests that none could realize on their own. The shared interest in the case of elevators is finding an acceptable compromise between two goals: using the elevator when it arrives, and avoiding unacceptable crowding. No elevator user can strike an acceptable balance on his or her own; others must cooperate by standing in appropriate places. Following the "maximize the distance from your nearest neighbor" norm creates the necessary cooperation.[16] Similar

14. Our notion of coordination norms is similar to but not as broad as Steven Hetcher's notion. *See generally* STEVEN A. HETCHER, NORMS IN A WIRED WORLD 50 (2004) (stating that coordination norms are "a pattern of rationally governed behavior maintained by a group in conformity" in order to derive a "coordination benefit," but they "need not be a proper coordination equilibrium."). Our notion is closely related to the notion of coordination game in game theory, which has roots going back to THOMAS C. SCHELLING, THE STRATEGY OF CONFLICT (1960) and to David Lewis' notion of convention. *See* LEWIS, *supra* note 13, at 36. There are important affinities between our notion of a coordination *norm* and the notion of coordination *game*. The original idea of coordination games and the term "coordination game" comes from Schelling. SCHELLING, *supra* at 89–90. The latter notion was further developed and connected to norms and conventions by Lewis. For a more recent treatment, see generally RUSSELL COOPER, COORDINATION GAMES: COMPLEMENTARITIES AND MACROECONOMICS (1999) (discussing macroeconomic examples of coordination).

15. This is a simplification. The true norm is closer to "maximize the distance from your nearest neighbor subject to the constraint that you stay within the peripheral vision of at least one other passenger and that you have at least one other passenger within your peripheral vision."

16. Following the norm is not of course a unique solution to the cooperation problem; there are alternatives—e. g., maximize the distance from your nearest neighbor and do not enter unless that distance is at least three inches.

points hold for driving on the right. No driver alone can realize the goal of everyone driving on the same side of the road. The norm ensures the necessary cooperation.

2. Why People Conform to Coordination Norms

A key claim in our analysis is that coordination norms resist change; once established, they are self-perpetuating. To explain why, we need to see why people conform to norms. We begin with *non*-coordination norms (like the Protestant church norm) and then turn to coordination norms.

People conform to non-coordination norms because, for the most part, people do what they sincerely and without reservation think they ought to do. Cases of "thinking one ought" form a continuum. At one extreme, one conforms only to avoid sanctions (one may avoid eating one's meat with one's salad fork only to avoid the disapproval of one's etiquette-obsessed friends); at the other extreme, sanctions play no role in explaining conformity. One conforms because one thinks that conformity realizes a state of affairs one regards as good—attending a Protestant church, for example. In between, conformity is a mix, in varying degrees, of both factors. People in Jone's town, for example, may attend church because they think it is their religious duty to do so and because others would disapprove if they failed to attend. Across the entire continuum, it is true to say that one thinks one ought to conform. The "ought" is a prudential "ought" at the "conform only to avoid sanctions" end, and a non-prudential "ought" at the "conform to realize a good state of affairs" end. Our free use of "ought" may ring false to those who assume that people are entirely self-interested.[17] We do not share the assumption, but those who wish to work within its constraints may simply interpret our "one ought to do" as "it is in one's self-interest to do." We will not make any claims inconsistent with that interpretation.[18]

We now turn to coordination norms. The explanation of why people conform to non-coordination norms is not adequate as an explanation of why they conform to coordination norms. To see why, recall that coordination norms are regularities that exist at least in part because everyone thinks that he or she ought to conform conditionally to the regularity. Thus, one will conform as long as one expects everyone else to do so. Our earlier explanation simply does not address cases in which one's convictions about what one ought to do depend on one's expectations about what everyone else will do. Our explanation of conformity to coordination norms is that conformity yields mutually concordant expectations about conformity, which yield conformity, which yields mutually concordant expectations about conformity, and so on.

---

17. The assumption dominates economics and law and economics. *See generally* AMARTYA SEN, THE IDEA OF JUSTICE 32–33 (2009); AMARTYA SEN, ON ETHICS AND ECONOMICS 15–28 (1987). Sen extensively criticizes the assumption, decisively in our view. *See id*. (arguing that rational actors do not exclusively act in their own self interest at all times).

18. Even our observation that one may conform to realize a good state of affairs is consistent as long as one sees such motivations as being, in one way or another, in one's self-interest.

In this way, once established, coordination norms are self-perpetuating. There are two questions. How does conformity yield expectations? And how do expectations yield conformity?

It is easy to see how conformity yields expectations. Imagine Alice is about to enter an elevator. Like anyone who has lived long enough in the community in which the elevator norm obtains, Alice knows that people conform to the norm because think they ought to.[19] What is true of Alice is true of everyone. Everyone who has lived long enough in the community knows that people conform because they think they ought to. Thus, mutually concordant expectations exist: everyone expects everyone to conform.

Now how do those expectations give rise to conformity? Start again with Alice. Alice thinks she ought to conform as long as everyone else does so, the expectation that everyone else will conform gives her good reason to conform when she enters the elevator, and acting on that reason, she will conform. Again, everyone is like Alice. Each person thinks he or she ought to conform as long as everyone else does, so the expectation that everyone else will conform gives each person a reason to conform, and acting on that reason, each will conform. Thus: conformity yields mutually concordant expectations about conformity, which yields conformity. The continuing conformity reinforces the mutually concordant expectations about conformity, which yield conformity, which reinforces the mutually concordant expectations about conformity, and so on. The process ensures that, once established, coordination norms are entrenched self-perpetuating practices. Our critique of software sales is that the "wrong" product-risk coordination norm has become entrenched in precisely this self-perpetuating way.

B. Value-optimal Norms

The product-risk norm governing software sales is "wrong" in the sense that it is not value-optimal. So what is a value-optimal norm? To answer, consider first that one typically conforms to norms without much thought; when you step into an elevator, you just unreflectively stand in the appropriate spot. You think you ought to stand there, but you do not worry or wonder about the justification for that "ought." You could justify it, however, if you reflected on the norm under ideal conditions (including having sufficient time, sufficient information, lack of bias, and so on).[20] You could justify the balance the norm strikes between not feeling crowded and being able to use the

19. For simplicity, we are suppressing interesting issues about the type and extent of knowledge required for the existence of a norm. *See* LEWIS, *supra* note 13, at 52–76 (examining the communal expectations required for a norm). Given our discussion, we may legitimately assume that the required knowledge conditions are fulfilled.

20. The appeal to reasoning under appropriate conditions to justify normative conclusions begins (at least) with Aristotle. *See* ARISTOTLE, NICOMACHEAN ETHICS 235 (1911) ("And to like and dislike what one ought is judged to be most important for the formation of good moral character: because these feelings extend all one's life through, giving a bias towards and exerting an influence on the side of virtue and happiness, since men choose what is pleasant and avoid what is painful."). For a modern exposition and defense of this approach, see STEPHEN L. DARWALL, IMPARTIAL REASON 201–17 (1983) (discussing the normative aspect of the reasoning behind a person's actions).

elevator when it arrives. Roughly speaking, a norm is value-optimal when one can, in light of one's values, justify the norm.

This is "rough speaking" because justification is a matter of degree. One might, for example, regard the elevator norm as justified but also think that the following alternative is even better justified: maximize the distance from your nearest neighbor and do not enter the elevator unless that distance is at least three inches. It is essential to take degrees of justification into account to arrive at an explanation of value-optimality that will serve our purposes in what follows. Thus, we define a value-optimal norm as follows: a coordination norm is value-optimal when (and only when), in light of the values of (almost) all members of the group in which the norm obtains, the norm is at least as well justified as any alternative. It is the "at least as well justified as any alternative" which makes the norm optimal; it means one cannot improve by choosing a *better* justified norm. There are many optimality notions; Pareto optimality is perhaps the most well-known one.[21] Value-optimality is the notion that we need for our analysis.

Our analysis of software vulnerabilities focuses on a particular type of failure of value-optimality. The following example illustrates the type. Until 1979,[22] hockey players in the National Hockey League did not wear helmets despite the clear risk of severe head injury.[23] There were two disadvantages to wearing a helmet: non-helmet-wearing players' perception that helmet-wearers lacked toughness, and a small loss in playing effectiveness against non-helmet-wearing players from the helmet's restriction of peripheral vision.[24] Nonetheless, had one conducted a secret ballot at the time, the vast majority of players would have agreed that it would be better if all players wore helmets.[25] "One player summed up the feelings of many: It is foolish not to wear a helmet. But I don't—because the other guys don't. I know that's silly, but most of the other players feel the same way."[26] In light of the sanctions, each player thought he ought to conform. The result was that it remained a norm not to wear a helmet until 1979, when the league required helmets.[27] Despite its persistence, the "no helmet" norm was not value-optimal. There was an

21. A situation is Pareto optimal when and only when it is not possible to improve the well-being of any one person without making others worse off. *See* ROBERT COOTER & THOMAS ULEN, LAW & ECONOMICS 14 (Sally Yagan et al. eds., 6th ed. 2011).

22. Andrew Podnieks, *How Fighting Became So Ferocious*, N.Y. TIMES (Dec. 15, 2011, 2:48 PM), http://slapshot.blogs.nytimes.com/2011/12/15/how-fighting-became-so-ferocious/.

23. *See* Thomas C. Schelling, *Hockey Helmets, Concealed Weapons, and Daylight Saving: A Study of Binary Choices with Externalities*, 17 J. CONFLICT RESOL. 381, 381 (1973) (discussing players' refusal to wear helmets absent a National Hockey League mandate, despite a fellow player's serious brain injury).

24. *Id*.

25. *The Economist* reports that there really was a secret ballot. *The Economics of Hockey Helmets*, ECONOMIST (July 19, 2007, 17:08), http://www.economist.com/blogs/freeexchange/2007/07/the_economics_of_hockey_helmet (citing James Surowiecki, *Fuel for Thought*, NEW YORKER (July 23, 2007), http://www.newyorker.com/talk/financial/2007/07/23/070723ta_talk_surowiecki). We have been unable to confirm this report. Thomas Schelling considers the results of hypothetical choices in THOMAS C. SCHELLING, MICROMOTIVES AND MACROBEHAVIOR 198–201 (2d ed. 2006), but he does not consider a secret ballot among hockey players.

26. Schelling, *supra* note 23, at 381 (citation omitted).

27. Podnieks, *supra* note 22.

alternative the players regarded as far better justified: all players wear helmets.

This example shows why value-optimality matters. The no-helmet norm defined a tradeoff between the risk of head injury on the one hand, and peripheral vision and appearing tough on the other. When they conformed to this norm, the players accepted this tradeoff—even though they regarded another norm (all players wear helmets) implementing a different tradeoff (reduced risk of head injury) as far better justified. This is why value-optimality matters: conformity to a norm that lacks value-optimality means acting contrary to one's values. We argue in Part V that software buyers are trapped in conformity to a product-risk coordination norm that lacks value-optimality.

C. Coordination Norms and Coordination Games

Our notion of coordination norm has strong connections to the notion of a coordination game in game theory.[28] This subsection is not essential to our argument, and readers with no taste for technical details may wish to skip the discussion. However, examining the connections with coordination games sheds important light on our use of the notion of a value-optimal norm. We assume some basic familiarity with game theory,[29] and we first briefly recall some standard definitions.

1. Definitions

In a (normal-form) game, each player has a finite set of actions available. Each player simultaneously chooses an action, and the outcome of the game, which is a distinct payoff to each player, is determined by the actions chosen. A player's strategy specifies what action he or she will use; a pure strategy is the choice of one particular action; a mixed strategy randomizes over two or more actions (e.g., if the possible actions are "Left" and "Right," then one mixed strategy is, "Choose Left with probability 1/3; Choose Right with probability 2/3"). A set of one strategy for each player is called a strategy profile. A strategy profile is a Nash equilibrium if each player's strategy is a best possible response to the combined strategies of all the other players.[30] Nash's famous theorem says that every game has at least one mixed-strategy Nash equilibrium.[31] Intuitively we would expect that a game with pure-strategy Nash equilibria that is played repeatedly will wind up with the players settling into one of those equilibria.

Normal-form games with only two players are typically described by giving a payoff matrix that shows the payoffs for all possible actions by the

28. *See* COOPER, *supra* note 14, at viii–x.

29. For a good overview of game theory, see, e.g., ROBERT GIBBONS, GAME THEORY FOR APPLIED ECONOMISTS 2 (1992); KEVIN LEYTON-BROWN & YOAV SHOHAM, ESSENTIALS OF GAME THEORY: A CONCISE, MULTIDISCIPLINARY INTRODUCTION 3 (2008); MARTIN J. OSBORNE & ARIEL RUBINSTEIN, A COURSE IN GAME THEORY 11 (1994); PHILIP D. STRAFFIN, GAME THEORY AND STRATEGY 63 (1993).

30. OSBORNE & RUBINSTEIN, *supra* note 29, at 216.

31. LEYTON-BROWN & SHOHAM, *supra* note 29, at 10.

players. For example, for the game of deciding which side of the road to drive on, with actions "Left" and "Right" we have:

	Left	Right
Left	(10, 10)	(0, 0)
Right	(0, 0)	(10, 10)

Figure 1: The Driving Game (which side of the road?)

In each cell of the matrix, there is a pair of numbers. The left number gives the payoff to the player who chooses the row, and the right gives the payoff to the player who chooses the column. Here there are two Nash equilibria, one where both players drive on the right and one where both drive on the left.

Let us say that a game is a coordination game if it has at least two pure-strategy Nash equilibria where all players choose corresponding actions, and no other pure-strategy Nash equilibria.[32] Our driving game represents the purest possible sort of coordination game: both players have the same payoffs for every combination of actions, and there are strict Nash equilibria for the action profiles consisting of corresponding moves.

2. Value-optimality and Nash Equilibria

To see the connection to value-optimality, consider the Stag Hunt Game.[33] Two hunters each have to decide whether to hunt stag together (neither can catch a stag alone) or hunt rabbits separately (which they can

32. To be more precise, we should have said "the actions of the game can be labeled so that it has Nash equilibria with the players choosing the corresponding actions," because two games that become the same when the actions (or players) are relabeled are really the same game. There does not seem to be any one exact definition of "coordination game" used uniformly throughout the literature. For instance, we did not specify whether the Nash equilibria must be *strict*, that is, whether "best response" in Nash equilibrium is to be defined as "better than all alternatives." (If it is defined as "at least as good as all other alternatives," then we get *weak* Nash equilibrium). Some authors impose further conditions that we will not discuss here.

We speculate that there is no precise definition because the notion of coordination is of most interest in social science and legal communities interested in social norms or situations of mixed cooperation and competition, and perhaps of less interest to the mathematical game theory community that tends to be the source of strict definitions. Schelling remarks in his Preface to the 1980 edition of THE STRATEGY OF CONFLICT, "I wanted to show that some elementary theory, cutting across economics, sociology and political science, even law and philosophy and perhaps anthropology, could be useful not only to formal theorists but also to people concerned with practical problems. I hoped too, and I now think mistakenly, that the *theory of games* might be redirected toward applications in these several fields. . . . [G]ame theorists have tended to stay instead at the mathematical frontier." SCHELLING, *supra* note 14, at vi.

33. *See generally* BRIAN SKYRMS, THE STAG HUNT AND THE EVOLUTION OF SOCIAL STRUCTURE 1–13 (2003) (describing the Stag Hunt game, which "is a prototype of the social contract").

easily catch on their own).[34] Stags provide a lot more food than whatever number of rabbits each could catch alone, and thus each prefers cooperating to hunt stags to hunting rabbits alone. Thus the payoff matrix might be:

	Stag	Rabbit
Stag	(10, 10)	(0, 3)
Rabbit	(3, 0)	(3, 3)

Figure 2: The Stag Hunt Game

Or perhaps if one hunter is likely to catch extra rabbits if the other hunter is (futilely) hunting a stag by himself, it might look like:

	Stag	Rabbit
Stag	(10, 10)	(0, 5)
Rabbit	(5, 0)	(3, 3)

Figure 3: Stag Hunt game with slightly different payoffs

The key point is that either way the choice "Rabbit, Rabbit" forms a Nash equilibrium.[35] Each player, if he believes the other player will choose rabbit, should himself rationally choose rabbit.[36] But why would a player believe that the other will choose rabbit when they both prefer stag? Distrust is a sufficient reason. Imagine hunting stag is more difficult and uncertain than hunting rabbits. Each will hunt stag only as long as he or she believes the other will.[37] As soon as one of them becomes convinced that the other will desert the stag hunt to catch rabbits, he or she too will abandon the stag hunt. Where there is insufficient trust, hunting rabbits will become the norm.[38] It will be a behavioral regularity, and—since it is a Nash equilibrium—each will think he

34. *Id.* at 1.
35. *See id.* at 3.
36. *Id.*
37. *Id.*
38. *See id.* (stating that rational players "need a measure of trust" to hunt stag instead of rabbits).

or she ought to conform as long as the other does.[39]

The norm is not value-optimal, however. There is another Nash equilibrium—stag-stag, and each player believes that that is the best outcome and hence believes that the coordinating behavior to achieve this outcome is value-optimal.[40] However, as long as a player believes that the other players are going to choose rabbit, then he believes that he too should choose rabbit.[41] Indeed, it would not merely be risky but outright foolish to choose stag if one knows that the other players will choose rabbit.

The rabbit norm traps the players in a suboptimal equilibrium. Our notion of a value-optimal norm generalizes this idea beyond the confines of game theory. We will argue shortly that the stag-hunt game corresponds to buyers' choices in buying vulnerability-ridden software, with the rabbit action corresponding to settling for defective software, and the stag action corresponding to demanding higher-quality software. Demanding higher-quality software is the value-optimal alternative, but buyers are trapped in the choice of defective software.[42] It is illuminating in this regard to return to the 1970s professional hockey players.[43]

For simplicity, we consider just two players, each representing some substantial fraction of all the hockey players. Here we could get two quite different games depending on just what assumptions we make about the hockey players' utility. First we might get exactly the same payoff matrix as the one in Figure 3 changing only the labels on actions from Stag, Rabbit to Helmet, Bare.

	Helmet	Bare
Helmet	(10, 10)	(0, 5)
Bare	(5, 0)	(3, 3)

Figure 4: Hockey Helmet Game (as Stag Hunt)

We obtain the payoff matrix in Figure 4 by assuming that players prefer helmets to bare heads; and prefer an advantage in winning to an even game; and prefer an even game to being at a disadvantage.[44] However, another

39. *See id*. at 10 ("[W]hat a rational player in the stag hunt will do depends on what the player thinks the other will do.").

40. *Id.* at 3.

41. *Id.*

42. *See infra* Part IV.

43. *See supra* notes 22–27 and accompanying text.

44. To be precise, we obtain the payoff matrix shown by assuming that having a helmet is worth 7 units of utility, being bare-headed is worth 0, having an advantage in winning is worth 5, being in a neutral position

plausible set of assumptions about the hockey players' preferences gives us the famous Prisoners' Dilemma game.[45] We need assume only that their preference for an advantage in winning the game is larger than any preference for wearing a helmet. For example, we might have:[46]

	Helmet	Bare
Helmet	(5, 5)	(0, 10)
Bare	(10, 0)	(3, 3)

Figure 5: Hockey Helmet Game (as Prisoner's Dilemma)

The two payoff matrices look quite similar, but there is a very important difference. For the Stag-hunt version of the hockey helmet game, there are in fact two Nash equilibria, and if everybody in the league is playing bare-headed, then we have a difficult but potentially solvable problem: how do we move to the other Nash equilibrium, or in our terms, how do we move from the norm that is not value-optimal to the norm that is value-optimal? However, in the Prisoner's Dilemma version, there is only one Nash equilibrium, the low-payoff one at Bare, Bare paying 3 to each player.

Our use of value-optimality generalizes this game-theoretic theme from coordination games to coordination norms. When a norm lacks value-optimality, there is (at least) one other alternative norm that is better justified; in game theoretic terms, there are—so to speak—(at least) two Nash equilibria, one of which all players prefer to the other. The "so to speak" qualification is essential. We have defined coordination games only for two-player interactions; the parties to the norms we consider typically number in the millions. Still, we think that the coordination games offer mathematically precise model that illuminates the broader phenomenon of value-optimality in the case of coordination norms.

II. PRODUCT-RISK NORMS

Typically, product sales are governed by (more or less) value-optimal

for winning is worth 0, and being at a disadvantage is worth -7, and that the preferences are independent and can be added.

45. For an excellent, detailed discussion of the prisoner's dilemma, see WILLIAM POUNDSTONE, PRISONER'S DILEMMA 101–31 (1992).

46. Figure 5's payoff matrix is based on the assumption that the utilities for an advantage in winning, an even game, and a disadvantage are respectively 10, 3, and -2, and the independent utility for having a helmet is 2.

norms. In Section IV, we argue that software sales are not governed by an appropriate value-optimal product-risk, and thus are an aberration from the typical pattern. In this section, we illustrate the typical pattern with three examples.

Each example is a coordination norm that allocates the burden of avoiding losses. Sellers bear whatever investment is required to produce norm-conforming products while buyers bear the risks of loss from using a norm-conforming product (unless those risks are assigned to the seller by other norms, by law, or by contract). We have deliberately chosen examples that may appear to govern the allocation of risks of unauthorized access due to software vulnerabilities. We argue in Part IV that they do not. The argument rests on the following point, which we emphasize in the discussion of the examples: one applies product-risk norms against a background of shared judgments about the proper allocation of risk in particular cases. The relevant shared judgments in the case of software sales assign the vulnerability-created risk of unauthorized access in ways inconsistent with the examples, and indeed in ways generally inconsistent with product-risk norms governing non-software sales.

One preliminary question remains: who are the parties to the norms? The answer may seem obvious—buyers and sellers; after all, they need to coordinate so that sellers offer what buyers want to buy. Further, if the norms are to allocate risks between buyers *and sellers*, how could both not be parties to the norm? It is indeed possible to represent product-risk norms as buyer-seller coordination norms; however, it is also possible, and simpler and more elegant, to model the norms as norms to which the only parties are buyers. The key point is that sellers design mass-software in response to sufficiently large groups of buyers; hence, no buyer can unilaterally ensure that his or her desired level of risk will be available—only a sufficiently large collective demand can accomplish that. Coordination via product-risk norms creates the required collective demand, to which profit-motive driven sellers respond. Since the profit motive is sufficient to ensure the sellers' response, there is no need to see the sellers as a party to the coordination norm.[47]

A. The "Fitness" Norm

The norm is that buyers "demand" products that are fit for the ordinary purpose for which such products are used. We use "demand" here in the following sense: "demand a fit product and (other things being equal)[48] refuse to buy an unfit product." We will use "demand" in this "demand and refuse to buy" sense throughout in our discussion of product-risk norms.

There is no doubt that the "fitness" norm is indeed a norm. The required

47. *See infra* Part III.

48. The "other things being equal" is merely to handle exceptions that do not matter for our purposes—e. g., a buyer may accept an unfit product if he or she has a non-standard use for it, or if the seller is a relative from whom the buyer believes he must not refuse.

regularity exists: buyers do demand fit products.[49] Moreover, they think they ought to do so—conditionally. As long as everyone conforms, non-conformity would mean unilaterally demanding an unfit product. The demand would go unfulfilled, and the non-conforming buyer would forego the purchase of the product. To the extent that doing so is unacceptable, the buyer will think he or she ought to conform. Of course, if enough buyers were interested in purchasing "unfit" products, seller would begin to offer them (other things being equal), and a new "fitness" norm would develop to govern those sales; products "fit" under the new norm would not be "fit" under the old one.

Varying notions of fitness are possible because "fitness" is determined by contextually-sensitive normative judgments. It could hardly be otherwise. Fitness depends on the type of product, the circumstances in which it is ordinarily used, the knowledge and skill of typical buyers, and the values in light of which buyers evaluate the product.[50] In a significant range of cases, there is sufficient overlap in values, use, knowledge, and skill that buyers converge on roughly the same judgments of fitness in particular cases. *Lindy Homes, Inc. v. Evans Supply Co., Inc.*[51] is an excellent example, even though it does not concern the fitness norm (at least not directly). The case concerns the Implied Warranty of Merchantability, which is asserted in Uniform Commercial Code section 2-314(2)(c).[52] Under that provision, a seller warrants that the goods are fit for the ordinary purpose for which such goods are used.[53] The task before the court was to determine fitness.

Lindy Homes used electrogalvanized sixpenny casing nails in cedar plywood siding.[54] Electrogalvanized nails rust when used in cedar; nails galvanized by a different process—"hot-dipped"—are far more rust-resistant, and the standard practice in the construction industry is to use hot-dipped nails in cedar.[55] When the electrogalvanized nails rusted, Lindy Homes sued the

---

49. The demand has a long history. As British common law responded to the rise of a market economy in the seventeenth century, it explicitly noted that the commercial custom and practice was to offer fit products. Such acknowledgments, moreover, are not confined to modern market economies; Ancient Roman law also notes the same custom and practice. *See* JAMES OLDHAM, ENGLISH COMMON LAW IN THE AGE OF MANSFIELD 79–205 (2004); Friedrich Kessler, *The Protection of the Consumer Under Modern Sales Law*, Part 1, 74 YALE L.J. 262. 263–64 (1974); George L. Priest, *A Theory Of The Consumer Product Warranty*, 90 YALE L.J. 1297, 1299–1302 (1981). The existence of the demand is consistent with spectacular failures to meet it. For example, in June 2010, in just a small fraction of the recalls that month, "McDonald's asked customers to return 12 million glasses emblazoned with the character Shrek. Kellogg's warned consumers to stop eating 28 million boxes of Froot Loops and other cereals. Campbell Soup asked the public to return 15 million pounds of SpaghettiOs, and seven companies recalled 2 million cribs." Lindsay Layton, *A Slew of Defective Products Leaves Consumers with 'Recall Fatigue,'* SEATTLE TIMES, July 3, 2010, http://seattletimes.nwsource.com/html/nationworld/2012268615_recallfatigue03.html.

50. U.C.C. § 2–314(2)(c) (2011).

51. *See* Lindy Homes, Inc. v. Evans Supply Co., Inc., 357 So. 2d 996, 999 (Ala. Civ. App. 1978) (finding no breach of implied warranty because buyer could not prove the nails were not "fit for the ordinary purpose for which such goods are used").

52. U.C.C. § 2–314(2)(c).

53. *Id*. The norm and the legal rule are not the same; people generally know and adhere to the norm while only the relatively legally sophisticated are aware of U.C.C. § 2-314(2)(c).

54. *Lindy Homes*, 357 So.2d at 998.

55. *Id*. at 999.

seller, Evans Supply, for breach of the implied warranty of merchantability.[56] The court held that the electrogalvanized nails were fit for the ordinary use made of them, a use that did not include their use in cedar.[57] The court relied on the industry-wide normative judgment it was "common knowledge in the trade that galvanized casing nails should not be used in exterior siding because a . . . 'hot-dipped' galvanized nail is proper in such a condition."[58]

We have not argued that the "fitness" norm is value-optimal, but we take the point to be sufficiently plausible that we may, for purposes of illustration, assume it is. We make the same assumption about the next two examples.

B. The "Negligent Design/Manufacture" Norm

The norm is that buyers demand products that do not, as a result of negligent design or manufacture, impose an unreasonable risk of loss on buyers who use the product as intended. The relevant regularity exists: buyers demand such products,[59] and moreover, buyers think they ought conditionally to demand such products. The argument is essentially the same as in the case of the "fitness" norm. A buyer who had an unusual use for a particular product might not care whether the intended uses of the product imposed an unreasonable risk of loss; however, as long as everyone else conforms, such a buyer will think he or she ought to conform. Non-conformity would mean unilaterally demanding a norm-deviant product; this demand would go unfulfilled, and the buyer would forego the purchase of the product. To the extent that going without is unacceptable, such a buyer will think he or she ought conditionally to conform. As with the fitness norm, if enough buyers were interested in purchasing "unreasonably risky" products for an alternate use, seller would begin to offer them (other things being equal), and a new "negligent design/manufacture" norm would develop to govern those sales; products not "unreasonably risky" under the new norm might still be "reasonably risky" for the range of uses governed under the old norm.

Applying a "negligent design/manufacture" norm requires making two context-sensitive, fact-specific judgments: one about unreasonable safety and one about negligent design or manufacture. *In re Sony BMG Music Entertainment* is an excellent illustration.[60] Part of its merit is that it concerns software. The example illustrates that the norms we discuss in this section do indeed govern *some* aspects of software; our claim, which we defend in Part IV, is that the norms do not apply to risks arising from software

56. *Id.* at 997.

57. *Id.* at 999.

58. *Id.*

59. People clearly do think that sellers ought not to offer products that, as a result of negligent design, impose an unreasonable risk of loss on buyers who use the product in the intended way. It is difficult to imagine anyone sincerely claiming that sellers ought to offer such negligently designed products, and indeed precisely the opposite conviction plays a central role in the development of products liability law. *See, e.g.*, Richard Wright, *The Principles of Products Liability*, 26 REV. LITIG.1067, 1070 (2007).

60. *Sony BMG Music Entm't*, FTC File No. 062-3019 (July 17, 2007), http://www.ftc.gov/os/caselist/0623019/index.shtm.

vulnerabilities.[61]

Between 2003 and 2005, Sony BMG Music Entertainment sold over 14 million music CDs containing one of two copy protection programs—XCP or MediaMax.[62] The programs allowed users to make only three physical copies of the CD, limited the ability to transfer files from the CD to other devices (including the iPod), allowed Sony to monitor users' listening habits, and were extremely difficult to uninstall.[63] Buyers were not given adequate notice of these aspects of the software.[64] Thus, using the CDs imposed the following largely undisclosed risks: interference with plans to make more than three copies, interference with plans to play files on other devices, and the invasion of privacy by monitoring buyers' listening habits. Buyers found these risks unreasonable:

> Once the public became aware of the [risks] . . . CDs distributed with [the software] . . . experienced a steep drop-off in sales within some market segments. . . . In addition, Sony BMG spent millions to settle the steady stream of lawsuits arising out of the . . . incident. Less quantifiably, the resulting backlash from artists and customers significantly damaged the reputations of Sony BMG and its parent corporations.[65]

The unreasonableness judgments are fact-specific, context-sensitive judgments about the number of times it is reasonable to expect to copy music from a CD to other devices, about what sorts of devices it is reasonable to copy to, and about the legitimacy of monitoring music listening habits.

Fact-specific, context-sensitive judgments are also the basis of the determination that Sony's actions were negligent. It is a standard practice in the music CD business to conduct a pre-release review of copy protection software to determine whether it works acceptably.[66] Sony BMG certainly had the resources to conduct such a review.[67] If it did so, it did so negligently; it should have discovered the flaws in the software.[68] Sony might instead have relied on the expertise of the suppliers of First4Internet (XCP) and SunnComm

---

61. The software in *Sony BMG* did contain vulnerabilities. *See* Deirdre K. Mulligan & Aaron K. Perzanowski, *The Magnificence of the Disaster: Reconstructing The Sony BMG Rootkit Incident*, 22 BERKELEY TECH. L.J. 1157, 1166 (2007). Our discussion focuses exclusively on other aspects of the software.

62. Complaint at ¶3, Sony BMG Music Entm't, FTC File No. 062-3019 (Jun. 28, 2007) (No. C-4195), *available at* http://www.ftc.gov/os/caselist/0623019/ 0623019cmp070629.pdf.

63. *See* Bruce Schneier, *Sony's DRM Rootkit: The Real Story*, SCHNEIER ON SECURITY (Nov. 17, 2005), http://www.schneier.com/blog/archives/2005/11/sonys_drm_rootk.html.

64. Mulligan & Perzanowski, *supra* note 61, at 1168.

65. *Id*. at 1168–69. As a result of the ensuing consumer outrage, Sony lost roughly $6.5 million in return fees and manufacturing costs alone. *Id.* at 1170.

66. *Id.* at 1179.

67. The resources to conduct such a review were available to Sony BMG from Sony Corporation of America, which has a 50% interest in Sony BMG whose holdings include Sony Electronics and Sony Computer Entertainment America. *Id.* Sony, along with Philips, owns the rights to the core DRM patents of Intertrust. In theory, at least, Sony BMG could have implemented a suite of better technical solutions. *See* Press Release, Sony Corp., Philips and Sony Lead Acquisition of Intertrust (Nov. 13, 2002), *available at* http://www.sony.com/SCA/press/021113.shtml.

68. Sony's "decision [to offer the CDs with the copy protection software] points to a culpable failure of internal procedures to safeguard against the wide-scale distribution of flawed protection measures." Mulligan & Perzanowski, *supra* note 61, at 1178–79.

(MediaMax), but such reliance would clearly have been culpable.[69] First4Internet's expertise was in content filtering technology, particularly the recognition of pornographic images; it had virtually no experience in copy protection technology.[70] SunnComm was no better. It began as a provider of Elvis impersonation services and had the lack of business savvy and technological insight to purchase a 3.5" floppy disk factory in 2001. It had virtually no relevant experience with copy protection software prior to entering the contract with Sony.[71]

C. The Least-Cost Avoider Norm

The norm is that, other things being equal, buyers demand products that assign the risk of a loss to the party that can most cost-effectively prevent or remedy the loss—the least cost-avoider.[72] Car buyers rather than sellers, for example, bear losses for failure to change the oil sufficiently often, since the buyers are in possession of the car and are the ones who can most easily keep track of mileage. Another example: in the case of refrigerators, sellers are liable for defects in the motor while buyers are liable for wear and tear on the shelves and doors. The least-cost avoider is the seller in regard to motor defects because it has more expertise and benefits from economies of scale; the buyer, on the other hand, is the least-cost avoider in regard to damage to the motor, doors, and shelves since the buyer may avoid damage simply by careful use.[73]

To see that the least-cost avoider norm really is a norm, consider that allocating risks to the least-cost avoider yields a net savings overall. Widely shared values dictate that, other things being equal, one should realize such savings when one can. Thus, everyone thinks he or she ought conditionally to conform. Non-conformity would mean unilaterally demanding something else; the demand would go unfulfilled, and the buyer would forego the purchase of the product. Thus, the requirements for the existence of a coordination norm are fulfilled. The required regularity exists—buyers demand products in which the risks of use are allocated to the least cost-avoider; moreover, the regularity exists because buyers think they ought conditionally to conform.

One applies the least-cost avoider in light of fact-specific, context-sensitive judgments. Applying the least-cost avoider norm requires making fact-specific, context-sensitive tradeoffs between the least-cost avoider bearing losses and potentially conflicting goals. The reason is that, under the norm, the least-cost avoider bears relevant losses, other things being equal. "Other things" are not "equal" when imposing losses on the least-cost avoider

69. *Id.* at 1179–80.

70. *Id.* at 1180.

71. *Id.*

72. *See generally* GUIDO CALABRESI, THE COSTS OF ACCIDENTS: A LEGAL AND ECONOMIC ANALYSIS 136 (1970); Ronald H. Coase, *The Problem of Social Cost*, 3 3 J.L. & ECON. 1, 11 (1960).

73. Alan Schwartz & Louis L. Wilde, *Imperfect Information in Markets for Contract Terms: The Examples of Warranties and Security Interests*, 69 VA. L. REV. 1387, 1398 (1983).

unacceptably conflicts with other goals.[74] The norm assigns a risk of loss to the least-cost avoider when and only when there are no unacceptable conflicts with other goals.

D. Norm-Implemented Tradeoffs

In each of the above examples, the norm implements tradeoffs among competing goals. A norm-conforming seller must make tradeoffs because the greater the seller's investment of time, effort, and money in creating norm-conforming products, the less is available for pursing other goals. The tradeoff for norm-conforming buyers comes from bearing the risk of using norm-conforming products. They must invest in precautions to avoid those losses and spend the time, effort, and money to recover from losses they fail to avoid. The more they invest, the less they have for other pursuits.

E. A New Definition of Value-Optimal Norms

These points about tradeoffs allow us, in the case of product-risk norms, to replace our earlier, general definition of value-optimality with one that is equivalent but more informative. The earlier definition was that a norm is value-optimal when and only when it is at least as well justified as any alternative. In the case of product-risk norms, we can replace this general criterion with: a product-risk norm is value-optimal when and only when the tradeoffs it implements are as least as well justified as any alternative. We argue in Part IV that the norm governing software sales is not value-optimal because there is an alternative norm that implements a better justified tradeoff.

Car buyers rather than sellers, for example, bear losses for failure to change the oil often enough, because the buyers are in possession of the car and are the ones who can most easily keep track of mileage. Another example: in the case of refrigerators, sellers are liable for defects in the motor, while buyers are liable for wear and tear on the shelves and doors. The least-cost avoider is the seller in regard to motor defects because it has more expertise and benefits from economies of scale; the buyer, on the other hand, is the least-cost avoider in regard to damage to the motor, doors, and shelves since the buyer may avoid damage simply by careful use.[75]

We turn now to the question: why—and in what sense—do norm-compliant sales ensure only acceptable risks? We begin with the "in what sense" part of the question. What do we mean by "acceptable"?

III. ACCEPTABLE RISK AND IDEAL TRANSACTION CONDITIONS

We define acceptable as follows: product-risk norms ensure that the

74. One may, for example, think that someone who commits an intentional tort should bear the losses he or she causes even if the victim is the least cost-avoider. *See generally* Wright, *supra* note 59, at 1099–1103.

75. *See* Schwartz & Wilde, *supra* note 73, at 1398 (using refrigerators as an illustration of how comparative advantage determines warranty content).

design and manufacture of a product imposes only acceptable risks (when and only when the norm is value-optimal). To see the rationale, suppose you had a choice between various norms. How would you choose? You would choose the norm (or in the case of ties, one of the norms) best justified in light of your values. Our definition of "acceptable" simply acknowledges this fact. But one may rightly object, what if there are not enough value-optimal norms? Product-risk norms are paired with *particular* risks. The "fitness" norm, for example, addresses the risk of using unfit products but not risk of using a negligently designed or manufactured product. Instead the "negligent design/manufacture" norm addresses that risk. Product-risk norms cannot ensure acceptable risks when there are significant risks that are not addressed by at least one value-optimal product-risk norm.

Our solution is to introduce the first of the two assumptions characterizing ideal transaction conditions. The first is that there is no significant risk that is not governed by at least one value-optimal product-risk norm. Call this *norm completeness*.[76] Norm completeness defines an ideal that practice only approximates. Practice tends to approximate norm completeness because sellers and buyers have exchanged products for centuries, and over the years, relevant value-optimal norms have evolved. In Part V, we argue that software sales are an aberration that falls unacceptably short of the ideal of norm completeness.

Norm completeness guarantees that enough product-risk norms exist, but it does not guarantee that *sellers will conform to the norms*. Indeed, product-risk norms would appear to make buyers an easy target for exploitation. Norm-conforming buyers typically do not investigate products in any detail; they simply take it for granted that products do not impose any unacceptable risks as a result of their design and manufacture. So would sellers exploit that fact to sell products that do impose such risks when doing so maximizes profits?

Our answer is to introduce the second assumption characterizing ideal transaction conditions: the assumption of a perfectly norm-competitive market (discussed below). When both assumptions hold, the profit-maximizing strategy is for sellers to conform to product-risk norms. It is the profit-maximizing strategy in practice—to the extent practice approximates ideal transaction conditions. Our argument adapts a well-known law and economics argument.[77] We begin with a summary of the argument: (1) whenever a business violates a norm, at least some consumers will notice; (2) consumers who detect a norm-violation will not, other things being equal, buy from norm-inconsistent businesses; (3) businesses are unable to discriminate between

76. We will, for simplicity, assume that consistency with norms is an all-or-nothing matter: a transaction is either entirely consistent, or entirely inconsistent. In practice, consistency is often a matter of degree. Similarly, in regard to value-optimality, we assume that one's values show either that one ought to act in accord with a given norm, or that one ought not. In practice, there may be open questions where one's values do not show that one ought to act in accord with the norm but also do not show that one ought not.

77. The argument is adapted from Alan Schwartz & Louis L. Wilde, *Intervening in Markets on the Basis of Imperfect Information: A Legal and Economic Analysis*, 127 U. PA. L. REV. 630, 640 (1979), which discusses the profit-maximizing strategy of businesses in more detail.

consumers who will, and those who will not, detect a norm-inconsistency; (4) therefore, in a perfectly norm-competitive market, the profit-maximizing strategy is for businesses to conform to norms.

A. Detecting Norm Violations

It is quite unlikely that norm-inconsistent products will escape the notice of every buyer. Awareness of norm-inconsistent products can come from news reports, magazine articles, books, consumer watch-dog groups,[78] negative publicity from consumer complaints, and litigation. This is not to make any claim about how many buyers detect norm-violations. It is the second assumption, formulated later, that includes such a claim.

B. Norm-Violation Detectors versus Norm-Inconsistent Sellers

When buyers detect norm-inconsistent sellers, they will not—other things being equal—buy from them. Consider that a norm is a regularity to which one thinks one ought to conform. Norm-violation detectors will therefore perceive a norm-inconsistent seller as not treating them as they ought to be treated. Other things being equal, buyers will purchase from sellers they perceive as treating them as they ought to be treated, not from those whom they perceive as not doing so.[79]

C. Sellers' Inability to Discriminate

If sellers could reliably discriminate between buyers who will, and those who will not detect a norm-inconsistency, they could remain norm-consistent in the case of inconsistency-detectors but violate norms for the rest. Sellers can in some cases spot those buyers that are likely to detect violations of norms. They can easily identify repeat customers who have objected to violations in the past, and it would not take too much research to identify a customer as, for example, the President of a consumer protection group like Consumer Reports. Such cases aside, when you walk into a retail store or order an item over the phone or online, nothing reliably signals whether you will detect norm-inconsistent behavior.[80]

D. The Profit-Maximizing Strategy

The final claim is that when sellers cannot discriminate between those who do and those who do not detect norm-inconsistencies, then *in a perfectly norm-competitive market*, the profit-maximizing strategy is to conform to

---

78. *See, e.g.*, Robert A. Hillman, *Online Boilerplate: Would Mandatory Website Disclosure of E-Standard Terms Backfire?*, 104 MICH. L. REV. 837, 853 (2006) (discussing the role of watchdog groups).

79. *See, e.g.*, J. R. Avrill, *Studies on Anger and Aggression*, 38 AM. PSYCHOL. 1145, 1149 (1983) (noting that violation of norms in an exchange provokes anger and may lead to the termination of the exchange).

80. You may, of course, reveal yourself as an inconsistency-detector if you explicitly insist on norm-consistent treatment, or if you detect and object to norm-inconsistent behavior.

product-risk norms; hence, rational, profit-motive driven sellers will do so. The assumption of a perfectly norm-competitive market is the second idealizing assumption. In Part V, we consider the extent to which software markets approximate this ideal.

When is a market perfectly norm-competitive? Two conditions must hold. The first is standard economic notion of *perfect competition*.[81] We define competition as perfect when and only when five conditions hold[82]:

1. *Profit-motive Driven Sellers*. Businesses seek to maximize profit.[83]

2. *Lack of Market Power*. Neither sellers nor buyers can individually control the price or determine the features of a product or service.[84]

3. *Homogeneous Products and Services*. The products and services involved in pay-with-data exchanges are quite diverse, but the homogeneity that matters for us is that they are all pay-with-data exchanges. The relevant similarity is in the mechanism of the sale, not the items sold. The argument we offer works for all pay-with-data exchanges, no matter what is exchanged, so we our references below to "products and services" are to any particular product or service involved in a pay-with-data exchange.

4. *Zero Transaction Costs*. Competitors may costlessly enter and leave the market, and buyers can costlessly switch from one seller to another.

5. *Perfect Information*. The perfect information requirement takes various forms.[85] Minimally, buyers and sellers know all prices. Most generally, all buyers and sellers are assumed know everything relevant to their production and consumption decisions.[86]

The second condition adds to the knowledge specified in (5). To formulate the condition, recall the point made above: buyers will, other things being equal, not buy from a seller who violates product risk norms. The second condition is that there are enough buyers who know when norm violations occur. More precisely: there are enough norm-violation-detecting buyers that a seller's gain from norm-inconsistent behavior is smaller than the loss which results when norm-violation-detectors buy from norm-consistent sellers. We will need a name for this requirement. Call it the *sufficient*

---

81. *See, e.g.*, JEFFREY L. HARRISON, LAW & ECONOMICS: POSITIVE, NORMATIVE AND BEHAVIORAL PERSPECTIVES 24–25 (2d ed. 2007) (discussing perfect and non-perfect competition).

82. Our definition follows a standard pattern. *See* WALTER NICHOLSON & CHRISTOPHER SNYDER, MICROECONOMIC THEORY: BASIC PRINCIPLES AND EXTENSIONS 415 (2012).

83. *Id.*

84. Definitions often substitute the requirement that there be a large number of sellers and buyers; the point, however, is to make the size of the market sufficient to ensure that no one seller or buyer has the power to set prices and determine features.

85. Some definitions of perfect competition omit any mention of perfect information. *See* Scott A. Beaulier & Wm. Stewart Mounts, Jr., Asymmetric Information about Perfect Competition: The Treatment of Perfect Information in Introductory Economics Textbooks 1–3 (Sept. 2008), *available at* www.scottbeaulier.com/Information_Version_2.doc. We include it in our definition because appeals to perfect information (and real world approximations to it) play a central explanatory role for us.

86. *See id*. at 4–5.

86. *See id*. at 4–5.

detection requirement.

Together perfect competition and sufficient detection entail that the profit-maximizing strategy is to be a norm-consistent seller. Perfect competition ensures that every norm-violation detecting buyer will buy from norm-consistent sellers, if at least one such seller exists. Sufficient detection ensures that there are enough norm-violation-detecting buyers that norm-inconsistent sellers lose more than they gain. Thus, the profit-maximizing strategy is to be a norm-consistent seller; hence, rational profit-motive driven sellers will be norm-consistent.[87]

E. Summary of the Product-Risk Norms Model

The model makes two idealizations. The first is the assumption of norm-completeness; the second is the assumption of a norm-competitive market. Norm completeness ensures that every purchase is governed by value-optimal product-risk norms; perfect norm-competitiveness ensures that rational, profit-motive driven sellers conform to the norms. When both assumptions hold, product sales are governed by norms that implement acceptable tradeoffs—tradeoffs to which buyers give free and informed consent. These assumptions define an ideal that is only approximated in practice. The closer practices come to the ideal, the more product sales involve acceptable tradeoffs to which buyers consent.

We argue next that software sales fall unacceptably far short of this ideal.

IV. APPLYING THE MODEL TO SOFTWARE VULNERABILITIES

In this section, we focus exclusively on the failure to approximate norm-completeness. We consider perfect norm-competitiveness in Part VI. Norm-completeness requires that every significant risk be allocated by at least one value-optimal norm. There are two ways to fail to meet this requirement. Norms may not exist, or existing norms may not be value-optimal. We claim that software sales exhibit the latter sort of failure: sales are governed by a norm that is not value-optimal. Our argument is divided into three parts. We first identify the norm. We then explain why, appearances to the contrary, the earlier examples of product-risk norms do not apply. Finally, we argue that the norm is not value-optimal.

A. The "Vulnerability-Ridden" Norm

The "vulnerability-ridden" norm is that buyers demand vulnerability-ridden software. The required regularity obtains—buyers do demand such software. It is commonplace to complain that buyers are unwilling to pay a premium for more secure software; they demand quick-to-market, cheap,

87. We assume that sellers, as members of the community in which the norm obtains, are aware of the norms and realize that they fail to meet buyers' demands when they fail to act in accordance with demand–unifying coordination norms. *See supra* note 19.

vulnerability-ridden software.[88] The explanation of the existence of the regularity is that buyers think that they ought conditionally to demand such software. Thus, the conditions for the existence of a demand-unifying product-risk norm are fulfilled: buyers regularly demand vulnerability-ridden software, and they do so at least in part because they think they ought conditionally to do so. To see that buyers think they ought conditionally to demand vulnerability-ridden software, divide buyers into three groups: buyers ignorant of the relevant risks, buyers who are aware of the risks but underestimate them, and those who are aware of the risks and accurately estimate them. In each group, buyers think they ought conditionally to demand vulnerability-ridden software, but they think so for different reasons.

Group One: Ignorance. Many buyers lack relevant information; security experts, consumer advocates, and those who make or seek to influence public policy may understand the risks involved in using vulnerability-ridden software, but many users have at best a minimal understanding.[89] Since they are unaware of the risks, the buyers do not see why they should pay a premium for more secure software; hence, they think they *ought* conditionally to demand quick-to-market, cheap, vulnerability-ridden software. The "ought" is conditional because a buyer would change his or her mind if all other buyers demanded secure software. An isolated demand for vulnerability-ridden software would go unmet.

Group Two: Underestimation. Buyers may be aware of the risks but underestimate them. "An amazingly robust finding about human actors . . . is that people are often unrealistically optimistic about the probability that bad things will happen to them."[90] Like buyers who are simply unaware of the risk, risk-underestimating buyers do not see why they should pay more for secure software, and thus think they ought conditionally to demand quick-to-market, cheap, vulnerability-ridden software.

Group Three: Compelling Reason. Even when buyers correctly estimate the risks, they will still think they ought to demand vulnerability-ridden software. Imagine Alice deciding whether to use the Adobe Acrobat Reader; she is well aware that the Reader has significant vulnerabilities,[91] but given the "vulnerability-ridden" norm, she has only two options: use the Reader or not.

---

88. *See* MARK G. GRAFF & KENNETH R. VAN WYK, SECURE CODING: PRINCIPLES & PRACTICES 28 (2003); Douglas A. Barnes, Note, *Deworming the Internet*, 83 TEX. L. REV. 279, 297–99 (2004).

89. *See* Bruce Schneier, *Insider Threat Statistics*, SCHNEIER ON SECURITY (Dec. 19, 2005), http://www.schneier.com/blog/archives/2005/12/insider_threat.html (reporting that among corporate employees, "[t]wo thirds (62%) admitted they have a very limited knowledge of IT Security" and "[m]ore than half (51%) had no idea how to update the anti-virus protection on their company PC"). *See also Consumer Labeling for Software Security*, SANS INST. INFOSEC READING ROOM, http://www.sans.org/reading_room/whitepapers/awareness/consumer-labeling-software-security_10 (last visited Jan. 17, 2012) (stating that naïve consumers are uninformed of the risks involved with insufficient computer security). Consumer awareness has increased over time. Tim Wilson, *Consumer Awareness Of Online Threats Is Up, Study Says*, DARK READING (Jan. 25, 2010, 8:54 AM), http://www.darkreading.com/security/vulnerabilities/222400407/index.html.

90. Christine Jolls, *Behavioral Economics Analysis of Redistributive Legal Rules*, 51 VAND. L. REV. 1653, 1659 (1998).

91. *See* Joel Yonts, *PDF Malware Overview*, SANS INST. (July 19, 2010), http://www.sans.org/security-resources/malwarefaq/pdf-overview.php (showing the history of vulnerabilities discovered in Adobe Acrobat Reader).

There is no third option of unilaterally demanding and receiving a less vulnerable Reader. She will think she ought to conditionally use the Reader as long as she is confident that she can take reasonable precautions to protect *herself* from unauthorized access. She realizes that, to the extent she transmits .pdf files to others who may not exercise the care she does, she imposes on them risks of unauthorized access by giving them yet one more occasion to use the Reader. But such third party risks have virtually no impact on her decision; given the extremely widespread use of the Reader, her decision not to use it would only yield an infinitesimal reduction in the risks to others.[92]

B. Why Not Fitness, Negligent Design/Manufacture, and Least-Cost Avoider?

But what about the three product-risk norms discussed earlier—fitness, negligent design/manufacture, and least cost-avoider? Why don't they apply to software sales? Let us examine the fitness norm first.

How can vulnerability-ridden software be fit? To see why it is fit, consider that fitness is determined, not by the opinion of software experts, but by contextually-sensitive judgments of software buyers. Software sales violate the norm only if those judgments classify the software as unfit. Software buyers share no such judgment. They demand quick-to-market, cheap, vulnerability-ridden software. One may rightly object that our buyers who correctly assess the risks of using vulnerability-ridden software may regard such software as unfit. However, even if they do, their judgment is, so to speak, inert. Buyers assessing risk correctly still think they ought conditionally to conform to the "demand vulnerability-ridden software" norm and hence conform to the norm. That is the norm that governs, not the "negligent-design-manufacture" norm—despite any judgment of unfitness that buyers who assess risk correctly may make.

Essentially the same points hold for the "negligent design/manufacture" norm. It may appear to apply because some vulnerabilities are clearly the result of negligent design.[93] Software "buffer overflows" are a good example of this. A buffer is a temporary location on a computer that a program uses to store information before it sends it to the CPU for processing.[94] Programmers can take effective steps to ensure that, before storing information in a buffer, the program checks to see if the capacity of the buffer is large enough to contain the information. To fail to do so is to create a buffer overflow vulnerability, which one can exploit to take over a computer and make it run

92. The third-party risks are, in the terminology of economics, *externalities*—effects of a decision on those who did not make the decision and whose interests were not taken into account in making the decision. *See* HARRISON, *supra* note 81, at 44–45.

93. We are *not* using "negligent" here in legal sense. We simply have in mind the non-legal use to mean "without sufficient attention." We discuss negligence as a tort in Part VI.

94. *See, e.g.*, Corey Pincock, *Secure Windows Initiative Trial by Fire: IIS 5.0 Printer ISAPI Buffer Overflow*, SANS INST. INFOSEC READING ROOM, http://www.sans.org/reading_room/whitepapers/win2k/secure-windows-initiative-trial-fire-iis-50-printer-isapi-buffer-overflow_190 (last visited Jan. 24, 2012).

programs one has written.[95] The consensus of software development experts is that, in a wide range of cases at least, it is negligent to create a buffer overflow vulnerability.[96] As computer security writers note, "[T]he existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections."[97]

Agreement on such instances of negligence is not, however, sufficient to show that software sales violate the negligent design/manufacture norm. The norm is that sellers do not offer products that, as a result of negligent design or manufacture, *impose an unreasonable risk of loss on buyers using the product as intended*. Unreasonableness is determined by shared judgments that allocate risks of loss between sellers and buyers; thus, to claim that vulnerability ridden software imposes unreasonable risks is to claim the shared judgment does so. Software buyers share no such judgment. The argument is exactly parallel to the argument in the case of the "negligent design/manufacture" norm. Buyers prefer quick-to-market, cheap, vulnerability-ridden software. Buyers who assess risk incorrectly regard the risks as unreasonable, but it does not matter whether they do or not. They still think they ought conditionally to conform to the "vulnerability-ridden" norm.

The case for thinking software sales violate the least-cost avoider norm may, at first sight, seem considerably stronger, for as we will argue shortly, software developers are the least-cost avoider for a wide range of losses arising from software vulnerabilities. It does not follow, however, that sales of vulnerability-ridden software violate the least-cost avoider norm. One applies the least-cost avoider norm in light of shared normative judgments that allocate the burden of avoiding the risk of loss. Thus, to claim that vulnerability-ridden software violates the best loss avoider norm is to claim that shared judgments allocate burden on software developers in a significant range of cases. As we argue below in Part IV.C, the opposite is true. Buyers demand quick-to-market, cheap, vulnerability-ridden software. It does not matter whether buyers who assess risk incorrectly judge software developers to be the best loss-avoids in some cases; they conform to the "demand vulnerability-ridden software" norm anyway.

Arguing that software sales do not violate the three product-risk norms given as examples does not show that sales do not violate other product-risk norms; however, the argument generalizes to other norms. Consider any product-risk norm that purportedly assigns the risk of at least some vulnerability to software developers. That claim will be inconsistent with the fact that buyers demand quick-to-market, cheap, vulnerability-ridden software.

---

95. *See* Aleph One, *Smashing the Stack for Fun and Profit*, PHRACK MAG., Aug. 11, 1996, *available at* http://www.phrack.com/issues.html?issue=49&id=14#article (noting that "over the last few months there has been a large increase of buffer overflow vulnerabilities being both discovered and exploited").

96. *See, e.g.*, Pincock, *supra* note 94 (noting that "[b]ecause buffer overflows begin with poor programming practices it is essential that vendors train their programmers to write secure code").

97. *CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')*, COMMON WEAKNESS ENUMERATION, http://cwe.mitre.org/data/definitions/120.html (last updated Sep. 13, 2011).

C. The "Vulnerability-Ridden" Norm Is Not Value-Optimal

A product-risk norm is value-optimal when and only when the tradeoffs it implements are as least as well justified as the tradeoffs implemented by an alternative norm. The "vulnerability-ridden" norm is not value-optimal because there is a better justified alternative. Under the current norm, buyers bear the risk of loss from unauthorized access resulting from vulnerabilities; the better justified option shifts a good part of that risk on to software developers.

The existence of a consensus on this point may seem surprising. It is difficult to obtain reliable data concerning losses, even in the case of readily quantifiable data such as the time, effort, and money involved in detecting unauthorized access, diagnosing its effects, removing malware that may have been installed, and lost productivity resulting from network malfunctions. This difficulty does not, however, prevent widespread agreement that the cost of unauthorized access runs in the billions of dollars a year.[98] While not all of these losses can be traced back to software vulnerabilities, vulnerabilities are nonetheless a significant factor,[99] and the consensus is that the cost of improving software development procedures to an extent that would significantly reduce vulnerabilities would be considerably less than the aggregate cost of unauthorized access mediated by vulnerabilities.[100] Software developers are—to a considerable extent—the least-cost avoider with regard to a wide range of vulnerabilities.

This conclusion is reinforced by considering losses that resist quantification, primarily: invasion of privacy, loss of trust, and anxiety from a sense of increased risk.[101] The assessment of a matter of making normative judgments about the desirability of competing policy goals—in particular, the goals served by keeping software costs down versus the value of trust, privacy, and a reduced sense of risk. To the extent one thinks a reduction in non-quantitative losses is worth an increase in software development costs, one has an additional reason to regard software developers as the least cost-avoiders over a wide range of cases.

We conclude that the "vulnerability-ridden" norm is not value-optimal. The solution is to replace that norm with a value-optimal norm. But what

---

98. *See* CATE, *supra* note 1, at 6–7.

99. *See* Schultz, *supra* note 1, at 1.

100. *See* GRAFF & VAN WYK, *supra* note 88, at 56; ROGER S. PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH 13–14 (2001); PONEMON INST., BUSINESS CASE FOR DATA PROTECTION: A STUDY OF CEOS AND OTHER C-LEVEL EXECUTIVES IN THE UNITED KINGDOM 2 (Mar. 2010), http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file/IBM%20Business%20Case%20for%20Data%20Protection%20UK%20White%20Paper%20FINAL6%20doc.pdf (noting that "C-level executives believe the cost savings from investing in a data protection program of £11 million is substantially higher than the extrapolated value of data protection spending of £1.9 million. This suggests a very healthy ROI for data protection programs"). The study is of course not a study of investment in *software development*, but the significant savings from protecting data on networks suggest that reasonable software development practices that reduced the incidence of vulnerabilities would save money.

101. *See* Alessandro Acquisti et al., *Is There a Cost to Privacy Breaches? An Event Study*, ICIS 2006 PROCEEDINGS, 1563, 1564–65 (2006), http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1215&context=icis2006.

exactly should the alternative norm be? The more software developers must invest to create norm-conforming software, the less is left over for other important goals, including promoting software innovation,[102] promoting the development of open source software,[103] and ensuring sufficient competitiveness among software sellers.[104] The less developers invest, the greater the risk of loss from unauthorized access, and hence the greater the investment buyers must make in avoiding those losses or recovering from them when they occur. The more buyers invest, the less they have for the wide variety of other goals they pursue. A value-optimal norm must define a best justified tradeoff among the competing goals.

V. BEST PRACTICES AND BEST-PRACTICES NORMS

We claim that the norm should be that buyers demand software developed following best practices. One immediate difficulty is that "[b]est practices has become an overused, underdeveloped catchphrase employed by industries and professions to signal an often unsubstantiated superiority in a given field."[105] Accordingly, our first step is to explain what we mean by "best practices." We then argue for the "buyers demand best practices software" norm.

A. Best Practices Defined

A best practice in a particular industry is a practice (method, process, or system) meeting two conditions. The first condition consists of two parts. Part one: with regard to one or more goals, there must be widespread agreement that it is highly desirable that those goals be achieved.[106] Call these goals the *best practice goals*. Part two: there must be widespread agreement that following the practice is a sufficiently reliable, sufficiently detailed means of meeting the best practice goals.[107]

An example is helpful. In the United Kingdom, the Electrical Safety Council promulgates best practices for electrical wiring.[108] The Council offers "a series of Best Practice Guides in association with leading industry bodies

---

102. *See* MICHAEL A. CARRIER, INNOVATION FOR THE 21ST CENTURY: HARNESSING THE POWER OF INTELLECTUAL PROPERTY AND ANTITRUST LAW 19–33 (2009) (emphasizing the importance of innovation).

103. On the importance of open source software, see Edward M. Corrado, *The Importance of Open Access, Open Source, and Open Standards for Libraries*, ISSUES IN SCI. & TECH. LIBRARIANSHIP, Spring 2005, *available at* http://www.istl.org/ 05-spring/article2.html.; David A. Wheeler, *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!*, http://www.dwheeler.com/ oss_fs_why.html (last updated April 16, 2007) (offering statistics to show that open source software can be a better option than proprietary software). Best practices appropriate for proprietary software might unduly constrain the development of open source software.

104. *See generally* CARRIER, *supra* note 102, at 304–12 (discussing competition in intellectual property).

105. Ira P. Robbins, *Best Practices on "Best Practices": Legal Education and Beyond*, 16 CLINICAL L. REV. 269, 271 (2009).

106. *Id.* at 291 (explaining that the first requirement requires that "those who attempt to discover or define a best practice must agree on the goal that the practice is intended to achieve").

107. *Id.* at 292 (discussing benchmarking as an objectively verifiable means of meeting best practice goals).

108. *See generally Best Practice Guides*. ELECTRICAL SAFETY COUNCIL (Jan. 24, 2012), http://www.esc.org.uk/industry/industry-guidance/best-practice-guides/.

for the benefit of electrical contractors and installers, and their customers."[109] The best practice goal is adequate safety, a goal widely regarded as highly desirable;[110] further, there is also widespread agreement that following the practices is a sufficiently reliable way to achieve that goal. The practices contain specific, detailed requirements for testing and installation. The best practices for electrical wiring, for example, require that the electrician determine whether the insulation resistance in electrical circuits is at least one megaohm; if not, equipment on that circuit must be disconnected, or 30 mA RCD protection must be installed.[111] Even a cursory survey of best practices reveals that they typically provide quite detailed advice. We defer our explanations of "sufficiently reliable," and "sufficiently detailed" to the discussion of the second condition.

To formulate the second condition, note that practices meeting the first condition implement tradeoffs between the best practice goal and a variety of other competing goals.[112] The reason is that following best practices typically requires an increased investment of time, effort, and money. Conforming to electrical wiring best practices, for example, requires various inspections and the installation of hardware upgrades.[113] This increases the cost of maintaining buildings, and the increased cost entails tradeoffs between safety and other goals. Increased wiring costs can, for example, affect the availability of low cost rentals.

Best practices for pharmaceutical company staffing and expenditure are another example. The tradeoff is between costs and discovering, developing, and distributing high-quality drugs at reasonable prices. This fact forms a key selling point for Best Practices, a company that licenses access to a database of best practices:

> By finding the optimal level of staffing and spending to achieve efficiency and effectiveness, companies can save money while maintaining a high-value medical affairs function [discovering, developing, and distributing high-quality drugs at reasonable prices]. Medical affairs leaders can use the information in this . . . document to learn how top companies find the optimal level of staffing and spending to achieve both efficiency and effectiveness in executing the

---

109. *See, e.g.*, ELECTRICAL SAFETY COUNCIL, SELECTION AND USE OF PLUG-IN SOCKET OUTLET TEST DEVICES 2 (Jan. 24, 2012), *available at* http://www.esc.org.uk/fileadmin/user_upload/documents/industry/best_practice/BPG8_10.pdf.

110. *See Statistics*, ELECTRICAL SAFETY COUNCIL, http://www.esc.org.uk/stakeholder/policies-and-research/statistics/ (last visited Jan. 17, 2012) (offering safety statistics).

111. 30 mA RCD protection (the British equivalent of CGFI switches) greatly reduces the risk of an electrical shock sufficient to cause arterial fibrillation, the main cause of death from electric shock. *See* ELECTRICAL SECURITY COUNCIL, BEST PRACTICE GUIDE 6 (2d ed. 2010), *available at* http://www.esc.org.uk/pdfs/business-and-community/electrical-industry/BPG1v2_web.pdf.

112. *See*, ELECTRICAL SAFETY COUNCIL, REPLACING A CONSUMER UNIT IN DOMESTIC PREMISES WHERE LIGHT CIRCUITS HAVE NO PROTECTIVE CONDUCTOR 5 (Mar. 2010), *available at* http:// www.esc.org.uk/fileadmin/user_upload/documents/industry/ best_practice/BPG1v2_web.pdf (advising that "where the customer is . . . not prepared to accept the cost or disruption of re-wiring . . . but still needs a new consumer unit [circuit breaker box], . . . the installer needs to carry out a risk assessment before agreeing to replace only the consumer unit").

113. *See id.* at 6 (discussing the need for risk inspection and testing).

mission of medical affairs.[114]

To trade costs against healthcare is, of course, to trade costs against the vast number of concerns and goals affected by the quality and availability of health care. A large number of similar examples can be found in the Best Practices Database in Improving the Living Environment.[115] The database provides access to "the practical ways in which public, private and civil society sectors are working together to improve governance, eradicate poverty, provide access to shelter, land and basic services, protect the environment and support economic development."[116]

Now we can state the second condition: the tradeoffs implemented by following the practices are at least as well justified as any alternative.[117] This is what makes the practices *best* practices. One cannot improve the tradeoffs by switching to alternative practices. Discussions of best practices do not explicitly offer this "at least as well justified" gloss on what makes best practices best.[118] In its discussion of pharmaceutical best practices the company Best Practices characterizes best practices as "optimal"; such practices yield "the optimal level of staffing and spending."[119] Another common characterization is "best in class"—a company adopts best practices by "measuring . . . functions, processes, activities, products, or services against those of [its] competitors and improving . . . [to match] the best-in-class . . . ."[120] To be optimal or best-in-class is, however, surely to be at least as well justified as any alternative. Whatever the language used, we take it to be clear that a best practice is one that is at least as well justified as any alternative; if there is a better justified alternative, the practice can hardly be best.

The "at least as well justified" requirement explains why we do not require best practices to be the *most* reliable way to achieve the consensus goals.[121] The best justified tradeoffs may sacrifice some reliability in the name of furthering other goals. The requirement also explains why—and in what sense—best practices are sufficiently detailed methods. Recall the Electrical Safety Council requirement that 30 mA RCD protection must be installed in

---

114. *Medical Affairs Staffing & Spend: Maximizing Value, Decreasing Cost*, BEST PRACTICES, LLC, http://www.best-in-class.com/bestp/domrep.nsf/products/medical-affairs-staffing-spend-maximizing-value-decreasing-cost (last visited Feb. 3, 2012).

115. BEST PRACTICES DATABASE IN IMPROVING THE LIVING ENVIRONMENT, http://www.bestpractices.org (last visited Feb. 3, 2012).

116. *Id*.

117. *Cf.* Robbins, *supra* note 105, at 291 (noting that to be a best practice, there can be only one way to accomplish the best practice goal).

118. *See supra* Part II.E (discussing value-optimal norms).

119. *Medical Affairs Staffing & Spend*, *supra* note 114.

120. ROBERT J. BOXWELL JR., BENCHMARKING FOR COMPETITIVE ADVANTAGE 30 (1994). The quote characterizes "benchmarking"; benchmarking is setting standards as a step toward adopting practices that realize them. *Id.* at 17. The practices are "best practices" if they are "best in class." "State of the art" is a similar characterization; as Robbins notes, Great Britain also uses the term best practices in the area of public management, defining a best practice as generally accepted "state of the art" approach. Robbins *supra* note 105, at 281 (citing Tessa Brannan et al., *Assisting Best Practice as a Means of Innovation*, 4 LOC. GOV'T STUD. 23, 24 (2008)).

121. *See supra* Part II.E. (discussing value-optimal norms).

electrical circuits with less than one megohm of resistance.[122] This requirement allows one to compare Electrical Safety Council practices to practices that require different combinations of cost and protection against electrical shock. In general, where one has a variety of competing goals, one will want to compare various tradeoffs among those goals to determine which tradeoffs are the best justified. Best practices must be sufficiently detailed to allow one to make those comparisons.

B. Summary of the Argument for the Best-practices Norm

We begin with a summary of the argument. (1) Best practices for software development exist, and software developers would significantly reduce vulnerabilities if they followed them. (2) Best practices make tradeoffs among competing goals, where the tradeoffs are at least as well justified as the tradeoffs implemented by alternative practices. Therefore (3), a "buyers demand best practices software" norm would be a value-optimal norm whose implementation would significantly reduce vulnerabilities.

Premise (2) follows from our discussion of best practices. As that discussion shows, best practices for software development—assuming they exist—make tradeoffs among relevant competing goals that are at least as well justified as alternatives. Relevant goals include, as we noted earlier, prompting innovation, promoting the development of open source software, and ensuring sufficient competitiveness.[123] Given (2), the conclusion in (3) follows since a product-risk norm is value-optimal provided the tradeoffs it implements are at least as well justified as any alternative tradeoffs. The only question then is whether best practices exist for software development. One could reasonably think that they do not. It is, after all, routine to observe, as leading security expert Eugene Spafford does, that software "is usually produced using error-prone tools and methods, including inadequate testing."[124] Such practices can hardly qualify as *best*. Our view is that this does not show that best practices do not exist; it merely shows that existing best practices are not followed.

C. Best Practices for Software Development

We begin with a specific example. It is a best practice to ensure that, before a program stores information in a buffer, it first checks to see if the amount of information is greater than the capacity of the buffer.[125] Failing to do so creates a buffer overflow vulnerability.[126] The practice meets the two requirements for being a best practice.[127] The first requirement is that there

122. BEST PRACTICE GUIDE, *supra* note 102, at 6.

123. *See supra* notes 102–04 and accompanying text.

124. Spafford, *supra* note 9, at 36.

125. *See* CAPERS JONES, SOFTWARE ENGINEERING BEST PRACTICES: LESSONS FROM SUCCESSFUL PROJECTS IN THE TOP COMPANIES 531 (2010) (explaining that the use of debugging tools to check for buffer overflow problems "are so common that usage is a standard practice and therefore would be classed as a best practice").

126. *See id.* at 514 (noting that "buffer overflows are common examples of vulnerabilities").

127. *See* Robbins, *supra* note 105, at 291–92 (explaining the requirements for best practices).

must be widespread agreement that it is highly desirable to realize a certain goal, and there must be widespread agreement that following the practice is a sufficiently reliable and sufficiently detailed means of meeting that goal.[128] As we noted earlier, there is consensus on the goal of reducing vulnerabilities by requiring a greater investment in software development.[129] The consensus is, first, that an increased investment in software development would reduce the number of vulnerabilities in software over a significant range of cases, and hence the losses from unauthorized access.[130] There is also widespread agreement that the practice—ensuring that the amount of information to be stored does not exceed the capacity of the buffer—avoids buffer overflow vulnerabilities.[131]

The second requirement is that the tradeoffs which the practice implements must be at least as well justified as any alternative.[132] It seems to be true, as the consensus is that the time, effort, and money needed to ensure the amount of information to be stored does not exceed the capacity of the buffer is far less than the losses thereby avoided.[133] There are many such examples. A vulnerability is just a particular type of defect, similar in principle to any other software defect, such as giving the wrong answer or crashing. The same high-level picture holds for both software defects in general and software vulnerabilities in particular: the amount depends very much on the design and programming practices used.[134] There is widespread agreement that one should ensure adequate overall management of the creation of the software, from first deciding what the behavior of the software should be, and then through designing it, writing it, and especially testing it.[135] There is also widespread agreement on how to write and design the actual computer programs ("code" in the language of programmers) that collectively are the software.[136] This includes, for example, such matters as the choice of appropriate data structures and algorithms, structuring the flow of control well, obeying abstraction barriers, and breaking the overall software into appropriate size pieces.[137] The techniques for developing sufficiently defect-free software are collectively known as software engineering.[138] How to write individual

128. *See id.* at 278.

129. *See supra* Part IV.C.

130. *See supra* Part IV.C.

131. *See* Pincock, *supra* note 94, and text accompanying note 94.

132. *See supra* note 117 and accompanying text.

133. Ethan Preston & John Lofton, *Computer Security Publications: Information Economics, Shifting Liability and the First Amendment*, 24 WHITTIER L. REV. 71, 135–36 (2002); Krerk Piromsopa & Richard J. Enbody, *Secure Bit: Transparent, Hardware Buffer-Overflow Protection*, IEEE TRANSACTIONS ON DEPENDABLE & SECURE COMPUTING, Oct.–Dec. 2006, *available at* http://www.cse.msu.edu/cgi-user/web/tech/document?ID=619; David Wheeler, *Secure Programmer: Countering Buffer Overflows*, DEVELOPER WORKS (Jan. 27, 2004), http://www.ibm.com/developerworks/linux/library/l-sp4.html.

134. *See* ANDERSON, *supra* note 4, at 15.

135. *See* ALBERT ENDRES & DIETER ROMBACH, A HANDBOOK OF SOFTWARE AND SYSTEMS ENGINEERING: EMPIRICAL OBSERVATIONS, LAWS AND THEORIES 74 (2003) ("Well-structured programs have fewer errors and are easier to maintain.").

136. *See id.* (discussing the most important requirements for writing programs).

137. *See id.* at 35 (explaining that software engineers have to devise solutions to problems and designing the solution is "the most challenging and most creative activity in software and systems engineering").

138. *See id.* at 1 ("Software engineering is the part of systems engineering that deals with the systematic

computer programs well, and the basics of software engineering are fairly well-settled subjects,[139] and should be known by competent software developers. For example, one can find many aphorisms summarizing these principals in a handbook of software engineering.[140] More importantly, the basics of how to construct good quality code and the basics of software engineering formed a significant fraction of the core (required) portion of the model computer science bachelor's degree curriculum jointly published by the two main professional societies for computer science in 2001.[141] Furthermore, most of that same material was also found in the earlier 1978 and 1991 versions of that model undergraduate curriculum, though of course some important details have changed as the field has evolved. Writing secure software also requires some additional knowledge. Some minimal training in writing secure software is a standard part of today's undergraduate curriculum for computer science majors,[142] but was not so common a decade ago. In general, a great deal is known about what sort of software development practices lead to fewer software defects, and what sort lead to more defects. One particular area of software engineering that has seen real progress in the past 20 years or so is testing.[143] There are a whole host of automated techniques for testing whether software under development contains errors, and use of these techniques significantly lower the defect rate in the final product. Failure to use any of these newer testing techniques leads to higher defect rates. It is common wisdom among experts in software development that all the proper attention to all the issues we have mentioned lead to lower defect rates, and various studies from over the years back up this common

development, evaluation, and maintenance of software.").

139. However, the choice of *which* software engineering methodology is the best one for managing various sorts of projects is contentious. In particular, there is debate about the relative merits of a traditional methodology called the Waterfall Model, with its origins in the late 1960s, versus various other methodologies, such as Spiral or Agile. *See, e.g.*, David L. Parnas, A Rational Design Process: How and Why to Fake it 3, *available at* http://www.cs.tufts.edu/~nr/cs257/archive/david-parnas/fake-it.pdf (criticizing the Waterfall Model); Kent Beck et al., *Manifesto for Agile Software Development* (2001), http://agilemanifesto.org/ (outlining the Agile Model).

140. *See generally* ENDRES & ROMBACH, *supra* note 135. A small sample of the sort of rules includes: Boehm's first law: "Errors are most frequent during the requirements and design activities and are the more expensive the later they are removed." *Id.* at 17. Dijkstra–Mills–Wirth law: "Well-structured programs have fewer errors and are easier to maintain." *Id.* at 74. Fagan's law: "Inspections significantly increase productivity, quality, and project stability." *Id.* at 100. Herzel–Myers law: "A combination of different verification and validation [i.e., testing] methods outperforms any single method alone." *Id.* at 107.

141. *See, e.g.*, ERIC ROBERTS ET AL., COMPUTING CURRICULA 2001: COMPUTER SCIENCE 17 (2001). Of the roughly 280 hours of "core" material listed here, perhaps half the core material in Programming Fundamentals, a third of the core material in Programming Languages, and almost all the core material in Software Engineering concerns the basics of good software development practices. Together those hours make up about a third of that core curriculum. A recent revision does not make significant changes from the point of view of the issues we consider here, except for adding some material on how to write secure software to the core. *See* LILLIAN CASSEL ET AL., COMPUTER SOC'Y OF THE INST. FOR ELEC. & ELEC. ENG'RS & THE ASS'N FOR COMPUTING MACH., CS2008, COMPUTER SCIENCE CURRICULUM 2008: AN INTERIM REVISION OF CS 2001 (Dec. 2008), *available at* http://www.acm.org/education/curricula-recommendations.

142. *See* ROBERTS ET AL., *supra* note 135, at 17 (2001). For examples of standard textbooks on software engineering, see, e.g., ROGER PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH (7th ed. 2009); IAN SOMMERVILLE, SOFTWARE ENGINEERING (9th ed. 2010).

143. *See* ANDERSON, *supra* note 4, at 829.

wisdom.[144]

In sum, there are software development practices that meet the conditions for being best practices.[145] First, there is a goal—reducing the number of vulnerabilities—and there is widespread agreement that the goal is desirable, and that following the practices is a sufficiently detailed, reliable way to achieve that goal. Second, there is widespread agreement, at least to some extent, that the tradeoffs implemented by following the practices are at least as well-justified than any alternative. The "at least to some extent" qualification acknowledges that there is some indeterminacy here. It is clear that the tradeoffs involved in following certain best practices—such as checking on adequate buffer size—are at least as well justified as any alternative, and it is clear in general that one or more combinations of the practices discussed above implement tradeoffs that are at least as well justified as any alternative. But it is not clear what those combinations are. Exactly what tradeoffs among the various competing goals are best justified is unclear; there are competing arguments for weighing various goals in various ways.[146]

In evaluating such tradeoffs, it is important to bear in mind an often overlooked limit on what best practices can achieve. Software is different from other engineered products in that sufficiently complex software *inevitably* has programming flaws.[147] In contrast, design flaws are not *inevitable* in, for example, refrigerators, batteries, and bridges even when they exhibit considerable complexity. Software alone combines complexity and inevitable flaws. Thus, no matter how much one invests in development procedures designed to reduce programming flaws, flaws—and perhaps vulnerabilities—will remain. There are two reasons for this.

First, most of engineering is governed by *continuous* mathematics, whereas software is governed by *discrete* mathematics.[148] Continuous mathematics includes the mathematics of the real numbers, which describe the

144. *See generally* Anthony Hall, *Seven Myths of Formal Methods*, 7 IEEE SOFTWARE 11, 11–19 (Sept. 1990) (discussing Praxis studies and the CASE project). *See also* I. J. Hayes, *Applying Formal Specification to Software Development in Industry*, SE-11 IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 169, 175–76 (Feb. 1985) (discussing the usefulness of software engineering techniques in some particular projects); Alan MacCormack et al., *Trade-offs between Productivity and Quality in Selecting Software Development Practices*, 20 IEEE SOFTWARE 78, 81–84 (Sept.–Oct. 2003) (comparing various software engineering techniques).

145. *See generally* JONES, *supra* note 125, at 7–10 (describing what qualifies a software development practice as a "best practice").

146. *See* CARRIER, *supra* note 102, at 22 (discussing the analogous issues that arise in the context of copyrights and patents; many of the concerns and competing arguments cross over).

147. As far back as the 1980s, a panel convened to study the issues with software for President Regan's Strategic Defense Initiative noted, "Simply because of its inevitable large size, the software capable of performing the battle management task for strategic defense will contain errors. *All systems of useful complexity contain software errors.*" STRATEGIC DEF. INITIATIVE ORG., DEP'T OF DEF., 19980819-140. EASTPORT STUDY GROUP: SUMER STUDY 1985. A REPORT TO THE DIRECTOR, STRATEGIC INITIATIVE ORGANIZATION 14 (1985), *available* at http://dodreports.com/ada351613 (emphasis added). Recently, Capers Jones noted that one goal of software engineering best practices is to increase the percentage of bugs removed prior to delivery from 85 percent to something that "approach[es] 99 percent," (*not* that it approaches 100%). JONES, *supra* note 113, at xxvi.

148. Eric Roberts, *Computers and Society*, *in* ENCYCLOPEDIA OF COMPUTER SCIENCE 1591, 1594 (Anthony Ralston et al. eds., 4th ed. 2000).

physics of motion and electricity.[149] Discrete mathematics includes the mathematics of the integers and of strings of letters.[150] For our purposes, the heart of continuous mathematics is the notion of a continuous function. The definition of continuous function is typically given in calculus classes using Greek lambdas and epsilons, but what a continuous function means to an engineer is that if, in a continuous system, you make a very small error in one of your inputs, the error in the behavior of your system must also be small. The discrete mathematics that governs software offers no such guarantees. An error in a single line of a million-line program can cause arbitrarily large errors. The second thing that makes software different from other engineered entities is that there is no way to "over engineer" for safety in designing software, as one can in designing many physical systems.[151] For example, if one wants to design a building to withstand 140 mile per hour winds, one can do the calculations about the necessary material strength, thickness, etc., to withstand 150 mile per hour winds, and then build according to those calculations to create an extra margin for safety. There are analogous things to do in many engineering situations, but not in the construction of software.

D. Developers Do Not Follow Best Practices

Software developers do not follow the best practices. As we noted earlier, software "is usually produced using error-prone tools and methods, including inadequate testing."[152] Creating buffer overflow vulnerabilities is a clear violation of best practices, but the vulnerability is still a common occurrence[153] and still ranks third on the SANS Institute's 2011 list of the top twenty-five most dangerous software errors.[154] Why don't software developers conform more closely to best practices? The answer lies in the behavior of buyers. Buyers are trapped in a self-perpetuating coordination norm under which they demand vulnerability-ridden software. In such a case, the profit-maximizing strategy for software developers is to be the first in the market to offer a particular type of software or an upgrade to existing software. Reducing vulnerabilities by following best practices requires a longer and more costly development process, so software developers avoid those practices.

VI. CONDITIONS FOR CREATING THE NORM

The solution is to create a value-optimal, best-practices norm governing software sales in a market which sufficiently approximates perfect norm-

149. *See* J. D. JOSHI, FOUNDATIONS OF DISCRETE MATHEMATICS 1–3 (1989) (explaining the differences between continuous and discrete mathematics).

150. Roberts, *supra* note 148, at 1594.

151. *Id.*

152. Spafford, *supra* note 9, at 36.

153. Jason Lam, *Top 25 Series – Rank 3 – Classic Buffer Overflow*, APPSEC BLOG (Jan. 24, 2012, 7:42 PM), http://software-security.sans.org/blog/2010/03/02/top-25-series-rank-3-classic-buffer-overflow#comments.

154. *Id.*

competitiveness. The closer the market approximates perfect norm-competitiveness, the more rational, profit-motive driven sellers conform to the norm. The existence of the value-optimal norm ensures the norm-governed sales implement acceptable tradeoffs, tradeoffs to which buyers give free and informed consent. We remark in passing that the norm would do more than just reduce the number of vulnerabilities; it would reduce the number of software defects generally. As we noted earlier, a vulnerability is a type of software defect, and following best practices reduces defects generally.[155] How can one best ensure that a value-optimal norm operates in a sufficiently norm-competitive market? We first consider ensuring a sufficiently norm-competitive market and then turn to creating the norm.

A perfectly norm-competitive market exists when and only when two requirements are fulfilled: perfect competition and sufficient detection. Current markets fall far short of both requirements.

A. Perfect Competition

The operating system market falls far short of the requirement of multiple sellers. Microsoft dominates, with a relatively small market share going to Apple and Linux (and in the future, Google's Chrome operating system could possibly acquire a share of the market as well).[156] There are also significant barriers to entry, as operating systems are very costly to develop and whether they will be adopted is uncertain.[157] In addition, operating systems are not sufficiently homogeneous; switching from one to the other involves significant costs.[158] These issues require detailed analysis in the context of antitrust and intellectual property law,[159] and that task lies outside the scope of our efforts here. In contrast to the operating system market, markets for software applications and utilities may sufficiently approximate perfect competition, but

---

155. *See supra* Part V.C.

156. *Operating System Market Share*, NETMARKETSHARE.COM http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8 (last visited January 27, 2012).

157. *See* David A. Wheeler, *More than a Gigabuck: Estimating GNU/Linux's Size*, http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html (last updated July 29, 2002) (estimating that "it would cost over $1 billion ($1,000 million – a Gigabuck) to develop this GNU/Linux distribution by conventional proprietary means in the U.S. (in year 2000 U.S. dollars)"). *See also* Amanda McPherson, Brian Proffitt & Ron Hale-Evans, *Estimating the Total Development Cost of a Linux Distribution*, LINUX FOUND. (Oct. 2008), http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html. The authors note that:

> In 2002, David A. Wheeler published a well-regarded study that examined the Software Lines of Code present in a typical Linux distribution. His findings? The total development cost represented in a typical Linux distribution was $1.2 billion. We've used his tools and method to update these findings. Using the same tools, we estimate that it would take approximately $10.8 billion to build the Fedora 9 distribution in today's dollars [2008], with today's software development costs. Additionally, it would take $1.4 billion to develop the Linux kernel alone.

Id.

158. *See* SHAPIRO & VARIAN, *supra* note 7, at 103–72 (claiming switching costs lead to customer lock-in).

159. *See generally* CARRIER, *supra* note 102, at 5–10 (detailing various legal impediments to innovation); William H. Page and & Seldon J. Childers, *Software Development as an Antitrust Remedy: Lessons from the Enforcement of the* Microsoft *Communications Protocol Licensing Requirement*, 14 MICH. TELECOMM. & TECH. L. REV. 77, 79 (2007).

as with operating system markets, we will put the question aside. For our purposes, we may assume perfect competition.

B. Sufficient Detection

Sufficient detection is the requirement that there are enough norm-violation-detecting buyers that a seller's gain from norm-inconsistent behavior is smaller than the loss that results when norm-violation-detectors buy from norm-consistent sellers. It may appear that this condition is not fulfilled. Typical consumers lack the expertise required to distinguish—by inspecting the software—between vulnerability-ridden software and software with significantly fewer vulnerabilities.[160] This is worrisome, as it potentially leads to a lemons market. We first explain the notion of a lemons market, and then consider whether a lemons market does in fact exist in regard to software vulnerabilities.

We explain a lemons market using a version of the "used car" example first employed by the economist George Akerlof in his seminal article, *The Market for Lemons*.[161] Suppose a town has 300 used cars for sale: 100 good ones worth $2,000, 100 so-so ones worth $1,500, and 100 lemons worth $1,000. Buyers cannot tell the difference between a good and bad car; thus, buying a used car means entering a lottery in which the buyer has a 1/3 chance of getting a good car, a 1/3 chance of getting a so-so car, and a 1/3 chance of getting a lemon.[162] The expected value of the purchase is $1,500. Rational buyers, thus, will pay only $1,500 for a used car; consequently, buyers who value their good cars at over $1,500 do not offer those cars for sale. Thus, the market now contains lemons worth $1,000 and not-so-good cars worth $1,500; the expected value of a used car drops to $1,250; consequently, buyers who value their cars above $1,250 do not offer them for sale. The process continues until only the lemons are left on the market.[163] In general, a lemons market exists when four conditions are fulfilled: (1) the products on the market vary significantly in the extent to which they have certain properties (the properties that make a car a lemon, for example), and buyers regard products with the properties in question as having less expected value than those without them; [164] (2) there is an asymmetry of information where buyers cannot discriminate between products with the properties and those without, but sellers can at least partially distinguish them; and furthermore, (3) there is no

160. *See generally* John R. Michener et al., *"Snake-Oil Security Claims" The Systematic Misrepresentation of Product Security in the E-Commerce Arena*, 9 MICH. TELECOMM. & TECH. L. REV. 211, 223 (2003); Bruce Schneier, *How Security Companies Suckers Us With Lemons*, WIRED (Apr. 19, 2007), http://www.wired.com/politics/security/commentary/securitymatters/2007/04/securitymatters_0419?currentPage=all.

161. George A. Akerlof, *The Market for "Lemons": Quality Uncertainty and the Market Mechanism*, 84 Q. J. ECON. 488, 489–92 (1970).

162. *See id.* at 489 (hypothesizing that individuals will buy "without knowing" whether they are buying a lemon).

163. *See id.* (stating that the "bad cars" will drive the good ones off the market).

164. One may wonder about the meaning of "significantly"; considerations we offer in Part VII explain and motivate the qualification.

reliable signal of quality (i.e., sellers with an excellent car have no way to reliably disclose this fact to buyers); however, (4) buyers know there is a mix of products on the market.

Are these four conditions fulfilled for software vulnerabilities? In answering this question, it is important to distinguish two markets: the market for *security software and systems*, such as firewalls, anti-virus software, or secure USB memory sticks, and the market for other sorts of mass-consumer software. Bruce Schneier has argued convincingly the former market is a lemons market.[165] Others have picked up on his claim and argued that it may also apply to *software that is (relatively) secure*—that is, software that is relatively free of vulnerabilities.[166] We are not so sure. While there are strong arguments that *security software* is a lemons market, it is unclear whether *secure* software is a lemons market. Conditions (2) and (4) are arguably fulfilled, but (1) and (3) are problematic. We first briefly review the arguments in favor of regarding (2) and (4) as fulfilled. Condition (2): Typical consumers do not have the expertise to distinguish by inspecting the software between secure and insecure software,[167] while the developers do know something about what production practices they are using. Condition (4): Buyers—or at least a significant portion of buyers—do know that the market contains both vulnerability-ridden and not so vulnerability-ridden software.[168]

Condition (1) requires (in part) that buyers regard vulnerability-ridden software as having less perceived expected value than similar software with significantly fewer vulnerabilities. At the moment, this is not true. Buyers are, on the whole, *not* willing to pay more for more secure software.[169] Our proposal in Part VII is designed to change this, but if all it does is change consumer preferences, it may simply contribute to the creation of a lemons market in software. Accordingly, our proposal will also suggest a mechanism for avoiding a lemons market.

Condition (3) requires that there do not exist any reliable signals that differentiate vulnerability-ridden from similar software with significantly fewer vulnerabilities. Typical consumers do not have the expertise to distinguish *by inspecting the software* between vulnerability-ridden software and software with significantly fewer vulnerabilities. Inspection is not, however, the only way to determine the extent to which software suffers from vulnerabilities. The general quality of the software is a moderately reliable

---

165. *See* Schneier, *supra* note 160.

166. *See, e.g.*, Barnes, *supra* note 88, at 292 (noting that "[a]s long as software is maintained as a trade secret, and development occurs behind closed doors, buyers have nothing more to go on than vague, unprovable assertions about quality and security (which are cheap to make)" and asserting a lemons market results); Hahn & Layne-Farrar, *supra* note 1, at 314 (suggesting the possibility of a lemons market where software developers offered software that varied in the degree of security).

167. *See* Schneier, *supra* note 160 (noting that many consumers cannot tell the difference between encrypted USB drive options and implying that vendors will know the actual security capabilities of its devices).

168. *See* Hahn & Layne-Farrar, *supra* note 1, at 302 ("Software and network security issues receive substantial press.").

169. *See supra* Part IV.A (discussing how market conditions create a demand for vulnerability ridden software).

signal of the extent to which it contains vulnerabilities. Vulnerabilities are a kind of flaw or defect in the software, and it is reasonable to assume that their occurrence correlates with the occurrence of other flaws, such as a tendency to crash or give wrong answers.[170] Indeed, it is routine not to distinguish sharply between defects and vulnerabilities. As security experts observe in a recent book, "Software *defects* are the single most critical weakness in computer systems. . . . [S]oftware defects lead directly to software exploit[ation]."[171] The correlation between vulnerabilities and defects is sufficiently strong that at least some buyers will infer that improperly functioning software is likely to contain significant vulnerabilities.[172] This signaling mechanism is far from perfect, but sufficient detection does not require that all or most buyers detect vulnerability-ridden software, just that enough do to impose losses on sellers who offer such software. Thus, there is very possibly a signaling mechanism that is strong enough to prevent a lemons market.

Our ultimate proposal in Part VII for changing the market to encourage the creation of secure software does not rely solely on this possible signaling mechanism to avoid a potential lemons market. We argue that the statute we propose, if adequately enforced, will ensure that condition (1) fails to hold. That condition requires that software products vary *significantly* in the extent to which they have vulnerabilities, and that buyers regard vulnerability-ridden software as having less perceived expected value than similar software with significantly fewer vulnerabilities. We explain and give a motive for the "vary significantly" provision, and we argue that it will not be fulfilled. Thus, our proposal avoids the problem of a lemons market.

While showing how to avoid a lemons market is important, it is not the main thrust of our statutory proposal. Our central claim is that the statute, if adequately enforced, will give rise to a best-practices norm. We argue in Part VII that, once the norm is in place, the sufficient detection assumption will be more or less true. There will be, in enough different situations, enough norm-violation-detecting buyers that norm-inconsistent sellers suffer losses.

---

170. Programs containing vulnerabilities are often developed in ways that violated programming laws of the sort identified in ENDRES & ROMBACH, *supra* note 135 (citing programming laws). Development practices that violate those laws frequently create a variety of defects in addition to vulnerabilities.

171. GREG HOGLUND & GARY MCGRAW, EXPLOITING SOFTWARE: HOW TO BREAK CODE 14 (2004) (emphasis added). These lines come at the end of an introductory section of the book that moves from discussing famous software defects that had nothing to do with security and attackers to discussing defects that constitute security holes. *Id.* Two examples of non-security defects the authors give are the NASA's 1999 Mars lander software failure, where a metric versus English units error caused the loss of the $165 million systemand the Denver International Airport automated baggage handling system fiasco. *See, e.g.*, SARA BAASE, A GIFT OF FIRE: SOCIAL, LEGAL, AND ETHICAL ISSUES FOR COMPUTING AND THE INTERNET 417 (3 ed. 2008); MICHAEL J. QUINN, ETHICS FOR THE INFORMATION AGE 362 (4 ed. 2010); *NASA's Metric Confusion Caused Mars Orbiter Loss*, CNN (Sept. 30, 1999), http://articles.cnn.com/1999-09-30/tech/9909_30_mars.metric_1_mars-orbiter-climate-orbiter-spacecraft-team?_s=PM:TECH (last visited Feb 16, 2011).

172. *See* HOGLUND & MCGRAW, *supra* note 171, at 10–14 (stating that software defects are the single most critical weakness in computer systems).

C. Creating the Norm

Creating the norm requires ensuring that the conditions for the existence of a coordination norm are fulfilled. (1) The relevant regularity must obtain: buyers must regularly demand best practices software; and (2) the regularity must exist at least in part because buyers think they ought to conform as long as everyone else does.[173] Assume, for the moment, a perfectly norm-competitive market. Then, it is in principle clear how to ensure these conditions are fulfilled: convince almost all buyers to demand best practices software—where they demand this, at least in part, because they think they ought to as long as everyone else does. Assuming the demand persists long enough, profit-motive driven software developers will—in a perfectly competitive market—begin to meet the demand. When they do, buyers will continue to demand, and the following regularity will be established: buyers demand best practices software. The regularity will exist in part because buyers think that they ought to conform as long as everyone else does. The conviction that they ought to conform will be reinforced by the fact that unilateral non-conformity will mean going without software the buyer wants.

The assumption of a perfectly competitive market is essential. It ensures that developers will respond to the buyer demand; without such a response the demand will almost certainly fade away. Buyers of Netbook computers could demand Netbooks with a processor equivalent in power to the Pentium 4 processor, but sellers will not meet the demand a processor that powerful—it currently generates too much heat to function in a Netbook. Netbook buyers will either cease to demand such a processor, or they will cease to purchase Netbooks. The latter option is unlikely in the case of software generally, so an unmet buyer demand will eventually simply fade away.

D. The Approximation Goals

To summarize, there are three goals: (1) convince buyers that they ought conditionally to demand best practices software and ensure that they do indeed demand it for that reason; (2) avoid the creation of a lemons market; and (3) once the norm exits, ensure that, in enough different situations, there will be enough norm-violation-detecting buyers that norm-inconsistent sellers suffer losses. Call these the *approximation goals*. Market forces will not achieve the approximation goals.[174] Buyers are trapped in the self-perpetuating "vulnerability-ridden" coordination norm; moreover, the persistence of the

173. *See supra* Part I.A (discussing coordination norms).

174. The market has given rise to vulnerability disclosure businesses. iDefense, for example, pays for information about the existence of vulnerabilities and communicates this information to its clients. *See* iDEFENSE CYBER INTELLIGENCE, THREAT INTELLIGENCE AND SECURITY, http://www.verisigninc.com/en_US/products-and-services/network-intelligence-availability/idefense/index.xhtml (last visited Jan. 17, 2012). This is not a general solution for consumers, who will not be willing to pay the significant charges that businesses like iDefense demand. *See The Law And Economics Of Software Security*, *supra* note 1, at 315–316. CERT (Computer Emergency Response Team) discloses vulnerabilities for a credit. *CERT/CC Vulnerability Disclosure Policy*, CERT, http://www.cert.org/kb/vul_disclosure.html (last visited Jan. 16, 2012). The disclosures are too technical for the average user, however.

norm ensures that the profit–maximizing strategy is to be the first in the market to offer a particular type of software or an upgrade to existing software, even if the software or upgrade is imperfect in a variety of ways, including having vulnerabilities. As long as buyers are trapped in the norm, they will not demand best practices software. Even those who understand the individual and social advantages of such software are unlikely to do so; a unilateral demand for best practices software simply falls on deaf ears. We conclude that legal regulation is required to achieve the two approximation goals.[175] The question, then, is what sort of legal regulation will best achieve the approximation goals.

VII. CREATING THE NORM THROUGH LEGAL REGULATION

We first consider common law negligence and products liability for defective design, as well as statutory proposals modeled more or less along the lines of those two common law doctrines.[176] We argue that these approaches clearly fail to achieve the approximation goals. We offer a statutory alternative built around the idea of best practices.

A. Negligence

A software developer is liable in negligence for losses resulting from vulnerability only if the vulnerability was the result of the software developer's failure to act as a reasonable developer would.[177] There are a number of difficulties in using negligence to regulate vulnerabilities in software;[178] we focus entirely on assessing how well it will achieve the approximation goals.

It is certainly possible that negligence cases could lead to the fulfillment of the approximation goals. Here is one possible scenario. Successful negligence claims against software developers yield a series of decisions that, other things being equal, it is negligent not to follow this or that best practice

---

175. In general, norms arise through custom, private agreement, or legal regulation. *See supra* text accompanying note 13. A best–practices norm is unlikely to arise by custom as long as buyers are trapped in the "vulnerability–ridden" norm. It is also unlikely to arise by private agreement. Mass market, standard form contracts typically disclaim liability for direct and indirect damages and place limits on any potential liability. *See* Priest, *supra* note 49, at 1347–49. Priest's article presents empirical results in support of the claim that the disclaimers in standard form contracts are best explained as an optimal allocation of the risk of product malfunctions between the seller and the buyer. *Id.*

176. *See, e.g.*, NAT'L RESEARCH COUNCIL, CRITICAL INFORMATION INFRASTRUCTURE PROTECTION AND THE LAW: AN OVERVIEW OF KEY ISSUES 50 (Stewart D. Personick & Cynthia A. Patterson eds., 2003) ("As a motivating factor for industry to adopt best practices, tort law can be a significant complement to standard-setting, because compliance with industry–wide standards is usually an acceptable demonstration of due care.").

177. *See* Geoffrey T. Harvey, *New SQM (Software Quality Management) Methods, Tools Raise the Bar for Software Developers*, 3 SOFTWARE QUALITY MGMT. MAG. (Apr. 2003), *available at* http://www.bregmanlaw.com/_assets/docs/New%20SQM%20Methods.pdf ("[A] developer who produces a product that contains problems, such as security vulnerabilities, may be accused of negligence if the developer's production practices fall below standard industry practices and if the problems could have been avoided by using standard industry practices.").

178. *See* Chandler, *supra* note 8, at 155, 175 (discussing that in a hypothetical lawsuit involving a victim of distributed denial of service a "plaintiff will likely have suffered only pure economic losses, a category of loss for which courts are reluctant to permit recovery in negligence").

(e. g., it is negligent to create a buffer overflow vulnerability).[179] The "other things being equal" rider acknowledges that a developer who can demonstrate the reasonableness of a departure from best practices will not be liable. On the basis of the series of decisions, courts and software developers both conclude that, other things being equal, it is negligent not to follow best practices for software development.[180] Publicity about the law suits, combined perhaps with advertising from best practice complaint developers, convinces almost all buyers that they ought to demand best practices software, and they begin to demand it for that reason. *All* software developers respond by following the practices. This eliminates worries about fulfilling the sufficient detection condition. Since *all* software is best-practices software, there is no need to detect software that is not. Thus the following regularity arises: buyers demand best practices software. Once the regularity is in place, unilateral non-conformity will mean going without software the buyer wants, and buyers will think that they ought to conform as long as everyone else does.

Each step in this scenario is problematic. It is hardly automatic that the successful completion of the first step (namely, successful negligence claims against developers) would result in developers actually following best practices. Developers would have to be convinced that the cost of doing so was less than the expected legal liability.[181] Even in that case, it is hardly plausible that *all* developers will follow best practices; irrational developers will not do so. Further, it is far from obvious that publicity and advertising would convince almost all buyers that they ought to demand best practice software and lead them to demand it on that basis. Our focus, however, is on the first step—the assumption that courts will hold generally that it is negligent not to follow best practices. This is unlikely to happen; hence, it is unlikely that the process will even get started.

The role of custom in establishing reasonableness makes it extremely unlikely that courts will do so. As the *Restatement* notes, "In determining whether conduct is negligent, the customs of the community, or others under like circumstances, are factors to be taken into account, but are not controlling where a reasonable man would not follow them."[182] The relevant customs for software development are not the best practices but the prevailing industry practices.[183] In theory, industry practices are just "factors to be taken into account, but are not controlling where a reasonable man would not follow

179. Another example is a "time of check to time of use" vulnerability. *See CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition*, COMMON WEAKNESS ENUMERATION, http://cwe.mitre.org/data/definitions/367.html (last updated Sept. 13, 2011); J. Craig Lowery, *A Tour of TOCTTOUs*, SANS INST. INFOSEC READING ROOM (Aug. 2002), http://www.sans.org/reading_room/whitepapers/securecode/tour-tocttous_1049. As a type of race condition, time-of-check to time-of–use vulnerabilities rank twenty-fifth on the SANS list of the top twenty-five most dangerous software errors. *CWE/SANS TOP 25 Most Dangerous Software Errors*, *supra* note 3.

180. NAT'L RESEARCH COUNCIL, *supra* note 176, at 50–53.

181. *See* HARRISON, *supra* note 81, at 308 (stating that when prevention efforts are more costly than tort liability, then paying for liability rather than prevention is more efficient for businesses).

182. RESTATEMENT (SECOND) OF TORTS § 295A (1965).

183. *See id.* § 295A cmt. b. (1965); David Owen, *Proving Negligence in Modern Products Liability Litigation*, 36 ARIZ. ST. L. J. 1003, 1038 (2004).

them."[184] In practice, however, it is difficult for a plaintiff to overcome the defendant's claim that it followed industry practice and hence proceeded reasonably.[185] This is not a defect in tort law; it is a sensible approach to assessing reasonable design choices for one who is in the business of designing products to sell for a profit. What practices should a software developer adopt when designing software for sale in the current market? Buyers demand vulnerability-ridden software and will generally not pay a premium for more secure software. The developer's competitors cater to this demand by offering relatively inexpensive, insecure software. A developer who invests too much in software development runs the risk of business losses. Software developers, just as much as buyers, are in the grip of the "vulnerability-ridden" norm. In a wide range of cases, developers will be able to make a convincing case that they acted reasonably.

The case may not always be successful, of course. Courts have rejected such reasonableness claims where the plaintiff has identified a readily available way to avoid the damage that the industry practices ignore. A classic case is *The T. J. Hooper*.[186] Two tugs, the *Montrose* and the *T. J.Hooper*, encountered a gale while towing barges.[187] The tugs and the barges sank.[188] The cargo owners sued the barge owners, who in turned sued the owner of the two tugs; the owner petitioned to limit his liability.[189] The court found that the tugs negligently unseaworthy because they lacked shortwave radios.[190] Had they been so equipped, they would have received reports of worsening weather; had they received the reports, they would have avoided the storm by putting in at the Delaware breakwater.[191] The case illustrates a familiar pattern in torts cases: (1) an activity imposes a significant risk of harm on third-parties, where (2) those engaging in and benefiting from the activity under invest in protecting the third parties; (3) the law responds by imposing on those engaging in the activity a duty to take reasonable steps to prevent harm to third-parties, where (4) other things being equal, a reasonable step is one that reduces expected damage to third-parties by an amount greater than the total cost of the step. Current software development practices certainly appear to fit this pattern. Software developers underinvest in software development by ignoring best practices, thereby producing vulnerability-ridden software that

184. RESTATEMENT (SECOND) OF TORTS § 295A (1965).

185. *See* Gideon Parchomovsky & Alex Stein, *Torts and Innovation*, 107 MICH. L. REV. 285, 292 (2008). (utilizing Sledd v. Washington Metropolitan Area Transit Authority, 439 A.2d 464 (D.C. 1981) (per curiam) as an example of this effect).

186. The T. J. Hooper, 60 F.2d 737 (2d Cir. 1932). Parchomovsky and Stein cite *Texas & Pacific Railway Co. v. Behymer*, 189 U. S. 468 (1903), as a similar case. Parchomovsky & Stein, *supra* note 185, at 293. The *Behymer* court does indeed note that what "is usually done may be evidence of what ought to be done, but what ought to be done is fixed by a standard of reasonable prudence, whether it is complied with or not." *Behymer*, 189 U.S. at 470. *Behymer*, however, concerns the sudden stopping of a train in circumstances in which the court found the sudden stop negligent. There is no suggestion that sudden stops in such situations were an industry practice.

187. *The T. J. Hooper*, 60 F.2d at 737.

188. *Id*.

189. *Id*.

190. *Id*. at 739.

191. *Id*. at 739.

imposes, in the aggregate, significant losses on buyers and society as a whole. Should tort law hold that not following best practices is negligent? It is unlikely that the courts will do so. There are two key differences between shortwave radios of the *T. J. Hooper* and software best practices.

The first is that the cost of shortwave radios was relatively small.[192] The cost of acquiring a radio did not put a barge owner at a competitive disadvantage; indeed, it arguably conferred one since the owner could offer lower risk transport at the same cost as competitors. This is a critical factor in making it unreasonable not to acquire a radio—*even in the market context at the time*. The second difference is that barge owners could easily make a rough and ready comparison between the cost of the radio and the expected losses avoided by its use.[193] The losses, when they do occur, can be huge; and, while the occurrence of violent storms is difficult to predict, their occurrence from time to time is certain. This is a key factor in justifying the holding of negligence. If the comparison was uncertain and controversial, it would be far less clear that owners acted unreasonably. In the case of software, the comparison is uncertain and controversial. As we noted earlier, it is clear that some combination of best practices implement best-justified tradeoffs among the relevant goals, but it is unclear and controversial what combinations those are. For both of these reasons, it is unlikely that the courts will hold that it is negligent not to follow best practices.[194]

B. Products Liability for Defective Design

A product is defective in design only when use of the product involves a

192. As the court notes, "[a]n adequate receiving set suitable for a coastwise tug can now be got at small cost and is reasonably reliable if kept up; obviously it is a source of great protection to their tows." *Id.*

193. *See id*. (implying the benefits clearly outweigh the small costs).

194. This may strike some as dubious, because as Thomas Smedinghoff notes, "[r]ecent case law . . . recognizes that there may be a common law duty to provide security, the breach of which constitutes a tort." Thomas J. Smedinghoff, *Defining the Legal Standard for Information Security: What Does "Reasonable" Security Really Mean?*, *in* SECURING PRIVACY IN THE INTERNET AGE, *supra* note 8, at 22. In support, Smedinghoff cites Wolfe v. MBNA Am. Bank, 485 F. Supp. 2d 874, 882 (W.D.Tenn. 2007), Guin v. Brazos Higher Educ. Serv. Corp., No. 05-668, 2006 WL 288483, at *4 (D. Minn. Feb. 7, 2006), and Bell v. Michigan Council 25 of the Am. Fed'n of State, County, and Municipal Emps., AFL-CIO, Local 1023, No. 246684, 2005 WL 356306, at *5 (Mich. Ct. App. Feb. 15, 2005). These cases certainly support Smedinghoff's cautious claim that recent cases recognize that there may be a common law duty to provide security, but no case suggests that it is negligent not to follow best practices. *Guin* holds that a laptop theft from a home was not foreseeable because the person in possession of it lived in a relatively safe neighborhood, had taken reasonable steps to prevent burglary. *Guin*, No. 05-668, 2006 WL 288483, at *4. *Wolfe* concerns the failure to verify the authenticity information in a credit card application taken by a telemarketer. *Wolfe*, 485 F. Supp. 2d at 879. *Bell* concerns the non-online theft of information from a labor union; the court held that "defendant did owe plaintiffs a duty to protect them from identity theft by providing some safeguards to ensure the security of their most essential confidential identifying information." *Bell*, No. 246684, 2005 WL 356306, at *5. Other recent cases demonstrate that the courts may be reluctant to expand negligence doctrine to create liability for contributing to unauthorized access. In *Forbes v. Wells Fargo Bank, N.A.*, 420 F. Supp. 2d 1018, 1021 (D. Minn. 2006), the court rejects negligence liability for a bank's role in permitting unauthorized access to information that could be used to commit identity theft; the court notes that the plaintiff did not allege any harm, just an increased risk of harm. *Id.* Standard tort law does not allow recovery for a merely increased risk of harm. *Banknorth, N.A. v. BJ's Wholesale Club, Inc*., 442 F. Supp. 2d 206, 216 (M.D. Pa. 2006), holds that even where there is a present injury to the plaintiff, the economic harm rule prevents recovery when the injury is merely economic. *Id.*

foreseeable and unreasonable risk of harm.[195] As with negligence, the role of custom in establishing reasonableness makes it unlikely that courts will hold that failing to follow best practices (or a defensible alternative to best practices) creates a foreseeable and unreasonable risk of harm.[196] Evidence of industry practices is relevant under both of the main tests used to determine defectiveness—the "risk-utility" test (a product is defective when its risk of harm exceeds its benefits), and the "consumer expectations" test (a product is defective when it fails to meet the reasonable expectations of consumers).[197] Defendants may seek to show that a product was not defective by introducing evidence that other sellers customarily use the same design.[198] For the same reasons given in the discussion of negligence, it is unlikely that the courts will hold that it is negligent not to follow best practices.

C. Statutes Closely Modeled on Negligence or Products Liability

The arguments above also apply to any statute modeled sufficiently closely on the common law requirements for negligence or products liability; indeed, the critique applies to any statute that incorporates a "reasonableness" requirement for software development where the courts will rely heavily on custom in interpreting that requirement.[199] We suggest a different statutory alternative, modeled on best practices, as the best way to promote the approximation goals. Our goal is not to define the statute itself but to define the task of creating it. Our brief discussion is a catalog of problems to be solved, not a list of solutions.

D. A Statutory Task

The statute would identify best practices and require that software developers either follow them, or to avoid liability, be able to demonstrate the reasonableness of their alternative practices. It is essential to implement this requirement in a way that allows developers reasonable flexibility in their choice of development methodologies; otherwise, the statute will excessively inhibit innovation. The statute could also delegate to a standard setting organization, like the Computer Security Division of the National Standards Institute (NIST)[200] or the American National Standards Institute (ANSI),[201] to

195. *See* Wright, *supra* note 59, at 1078.

196. DAVID G. OWEN, PRODUCTS LIABILITY LAW 10–23 (2005) (discussing the use of custom in providing defectiveness derives at least in part from its use in proving negligence).

197. Parchomovsky & Stein, *supra* note 185, at 299.

198. David Owen, *Proof of Product Defect*, 93 KY. L.J. 1, 5 (2004). Plaintiffs may also seek to show that the product was defective by introducing evidence that other sellers use a safer design, but unless these other sellers are following best practices, this will not provide a basis for requiring that software developers follow best practices. *Id.*

199. Such statutory reasonableness requirements are common. As Smedinghoff notes in regard to statutory standards network security for businesses, "Laws and regulations rarely specify the security measures a business should implement to satisfy its legal obligations. Most simply obligate companies to establish and maintain 'reasonable' or 'appropriate' security procedures, controls, safeguards, or measures, but give no further direction or guidance." Smedinghoff, *supra* note 194, at 23.

200. COMPUTER SECURITY DIVISION, http://csrc.nist.gov/index.html. Computer Security Division does

adopt and enforce its standards. It could also delegate to an agency to fashion standards with advice from the private sector. There are well-known problems with the delegation approach.[202] Regulatory capture in particular is a concern. The concern is that commercial interests that dominate the software industry will have such a powerful influence on the formulation of the standards that the standards will fall far short of genuine best practices, and instead advance commercial or special interests.[203]

We now turn to issues that arise in using such a statute as a means to realizing the approximation goals.

1. Avoiding a Lemons Market

How would the statute ensure that there are a sufficient number of norm-violation-detecting buyers? As we noted in the discussion of negligence, the problem disappears if *all* developers follow best practices. This is of course extremely unlikely, but it is also not required. It is enough if *almost* all follow best practices. The main problem then is to ensure sufficient compliance.

But this may seem obviously wrong. If some developers deviate from best practices, will not the conditions for a lemons market arise? Recall that a lemons market exists when the following conditions hold: (1) the products on the market vary significantly in the extent to which they have certain properties (vulnerabilities, in this case), and buyers regard products with the properties in question as having less expected value than those without them; (2) there is an asymmetry of information where buyers cannot discriminate between products with the properties and those without, but sellers can at least partially distinguish them; and furthermore, (3) there is no reliable signal of quality;

---

not currently offer best standards for software development.

201. ANSI, http://www.ansi.org/ (last visited Feb. 3, 2012), does not currently offer standards for software development. It refers to the International Standards Organization ISO/IEC 24773:2008 standard. *New ISO/IEC International Standard to Certify Software Engineering Professionals*, ANSI, http://www.ansi.org/news_publications/news_story.aspx?menuid=7&articleid=2034 (last visited Feb. 3, 2012). ISO/IEC 24773:2008 does not specify best practices in our sense of the term. Its purpose is to establish "a framework for comparison of schemes for certifying software engineering professionals. A certification scheme is a set of certification requirements for software engineering professionals. ISO/IEC 24773:2008 specifies the items that a scheme is required to contain and indicates what should be defined for each item." INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, http://www.iso.org/iso/catalogue_detail.htm?csnumber=41543 (last visited Feb. 3, 2012). Other certification proposals have also failed to create viable best practices standards. Two notable failures are the Trusted Computer System Evaluation Criteria (TCSEC) (for a useful summary and links, see *Rainbow Series*, WIKIPEDIA, http://en.wikipedia.org/wiki/Rainbow_Series (last visited Feb. 3, 2012), and the COMMON CRITERIA, http://www.commoncriteriaportal.org/ (last visited Feb. 3, 2012). For criticisms of both approaches, see ANDERSON, *supra* note 4, at 517–38.

202. CARRIER, *supra* note 102, at 323–44 provides a succinct overview of the concerns. The discussion concerns standards in the sense of "a common platform that allows products to work together." *Id.* at 323. Essentially the same issues arise in defining best practices, however.

203. *See* Jon D. Hanson & David G.Yosifon, *The Situation: An Introduction to the Situational Character, Critical Realism, Power Economics, and Deep Capture*, 152 U. PA. L. REV. 129, 202–30 (2003) (explaining the concept of deep capture and its effects on the economy). The Federal Communications Commission is arguably an example of regulatory capture. *See* Hannibal Travis, *Of Blogs, eBooks, and Broadband: Access to Digital Media as a First Amendment Right*, 35 HOFSTRA L. REV. 1519, 1523–26 (2007) (describing the FCC's control over the broadcasting market). *See also* JONATHAN E. NUECHTERLEIN & PHILIP J. WEISER, DIGITAL CROSSROADS: AMERICAN TELECOMMUNICATIONS POLICY IN THE INTERNET AGE 392–95 (2005).

however, (4) buyers know there is a mix of products on the market.[204]

We suggested earlier that (3) probably does not hold, but we will not rely on that suggestion here. Instead, we note that (1) is most likely *not* fulfilled. If *almost* all developers offer best-practices software, the probability of purchasing non-best practices software is very low; hence the existence of such software on the market only minutely affects the expected value of a purchase. Rational buyers will simply ignore the minimal impact. They will not calculate the reduction in expected value of a purchase caused by the existence of vulnerability-ridden software. The reason is that it is rational not to try to assess the small difference in expected value and simply treat all software as if it were best-practices software. The costs of making the assessment are greater than any gain it yields.[205] Thus, half of (1) will be true: buyers will regard vulnerability-ridden software as having less expected value than software with significantly fewer vulnerabilities. But half of (1) will be false: the products on the market vary *significantly* in the extent to which they have vulnerabilities. There will not be enough vulnerability-ridden software to make it rational to take the minimal reduction in expected value into account in purchasing decisions.

2. Creating the Norm

How will the statute help convince buyers that they ought conditionally to demand best practices software and ensure that they do demand it for that reason? Our answer is that steps must be taken to "educate" buyers about the advantages of best-practices software. We put "educate" in quotes because techniques for creating the conviction form a continuum from genuine education to manipulation. At the "education" end, one presents the relevant information about the individual and social gains from more secure software and counts on rational reflection to create the conviction. As one moves toward the "manipulation" end, one increasingly supplements presentation of information and rational reflection with techniques designed to produce the conviction in other ways. The task is to use some combination of techniques to produce the desired conviction in buyers. One possibility is that, in order to gain a competitive advantage, developers who comply with the statute might themselves inform consumers about the advantages of "best practices" software and tout their software not only as best-practices software, but as software that exceeds the legal minimum. Alternatively, there are governmental ways to change citizens' minds, as the anti-littering, anti-smoking, and anti-drug campaigns illustrate.

---

204. *See supra* Part VI.B.

205. *See* A. M. Odlyzko, *The Case Against Micropayments*, LECTURE NOTES IN COMPUTER SCI. 2742 at 1, 5 (2003) (explaining that even small barriers to usage discourage usage). *See also* A. M. Odlyzko, *Internet Pricing and the History of Communications*, 36 COMPUTER NETWORKS 493, 501 (2001) (supporting the idea that fixed fees are more consumer-friendly).

3. Once the Norm is Established

Once the norm is in place, it is important that buyers and software developers conform on their own initiative, not because of the threat of enforcement of statutory requirements; otherwise, one must rely on difficult, costly, and uncertain enforcement. Developers will voluntarily conform as long as there is a sufficient number of norm-violation-detecting buyers. Once the norm is in place, there may well be. Developers themselves can ensure that buyers possess information about norm-inconsistent sellers. If Microsoft, for example, offers norm-inconsistent software, Google's advertising for its Chrome operating system can call that fact to buyers' attentions. Awareness of norm-inconsistent software can also come from publications like *Consumer Reports*, consumer watch-dog groups, and negative publicity from consumer complaints and litigation.[206]

VIII. CONCLUSION

Software sales currently depart dramatically from the typical pattern of sales governed, more or less, by value optimal-norms in a relatively norm-competitive market. Instead, buyers and developers are trapped in the "vulnerability-ridden" norm in a market that (most likely) falls far short of the norm-competitive ideal. The solution is to devise a suitable statutory stepping stone toward a value-optimal best-practices norm governing software sales in a sufficiently norm-competitive market.

206. *See* Hillman, *supra* note 63, at 853 (discussing the pros and cons of watchdog groups).