Towards a Model of the Costs of Security

David Larochelle
Department of Computer Science
University of Virginia
larochelle@cs.virginia.edu

Nicholas Rosasco
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
nrosasco@acm.org

Abstract

We present a simple information security model to determine why, historically, the level of security has not increased despite numerous technical advances. In our model, the software design process involves trade-offs between security and functionality. Developers choose points in the design space corresponding to certain levels of security and functionality. If development resources, such as number of developers, time for completion, etc., are fixed, there is an implicit trade-off between security and functionality. We refer to the set of points that represent the maximum possible security given a certain level of functionality as the protection possibilities frontier (PPF). Technical advances push back the PPF expanding the set of accessible points in the design space potentially allowing both increased security and increased functionality. But historically this has not been sufficient to result in increased security. Instead almost all of the technical advancement is used to increase functionality. We examine how technical advances affect the marginal cost of security in terms of sacrificed functionality and classify technical advances into 3 categories: security neutral, security hostile, and security enhancing. In order for the level of security to increase, security enhancing technical advances must offset security hostile technical advances. We also observe that producing security enhancing technologies is surprisingly difficult. Even advances in information security technologies often result in the ability to take additional risks rather than increased security. Additionally we briefly examine user preferences, which to a large extant drive the actions of developers. We suggest that the lack of security cannot be explained purely by consumer apathy and that limited product availability and network externalities also contribute.

1. Introduction

"the most secure computer in the world is one that has its disk wiped, is turned off, and is buried in a 10-foot hole filled with concrete. Of course, a machine that secure also turns out to be useless." -- [Viega02, 35].

As is the case in many areas, in information security there is no free lunch. The process of system design involves trade-offs between numerous factors. Developers and administrators have limited resources and multiple criteria to fulfill. To an extent, security is inherently opposed to functionality, as the above quote makes clear. But the trade-off is made more stark because projects are conducted with limited resources. There are a limited amount of developers, time for completion, and other resources. In order to produce software with greater security, there will be a cost such as increased development resources or less functionality in the product. For example, it may take longer to finish a more secure product, or it may be necessary to recruit an outside security expert.

We present a simplified model of software development that formalizes the above trade-off. In this model, developers have a finite amount of resources such as staff, equipment, and time, and they must choose between functionality and security. We define functionality broadly to include total development time and cost as well as features.

We depart from the traditional view that security is a binary property. Instead, we view security as a relative property. Although security cannot currently be precisely measured, and no commonly accepted measure of a program's security exists, it is common for programs and systems to be referred to as more or less secure. Judgments about the relative security of systems are usually based on past security flaws and the extent to which the developers emphasized security. Although these judgments are based on imperfect information and are often questionable, the idea that two systems contain different levels of security though neither has perfect security is consistent with the state of the practice.

2. Model

Figure 1 presents a graphical view of our model of the design space. The shaded area represents accessible points in the design space. The line represents the optimal points, which provide the maximum level of functionality for a given level of security. We refer to this line as the protection possibilities frontier (PPF). For example, Y represents a high level of security and a low level of functionality while Z represents a low level of security and a high level of functionality.

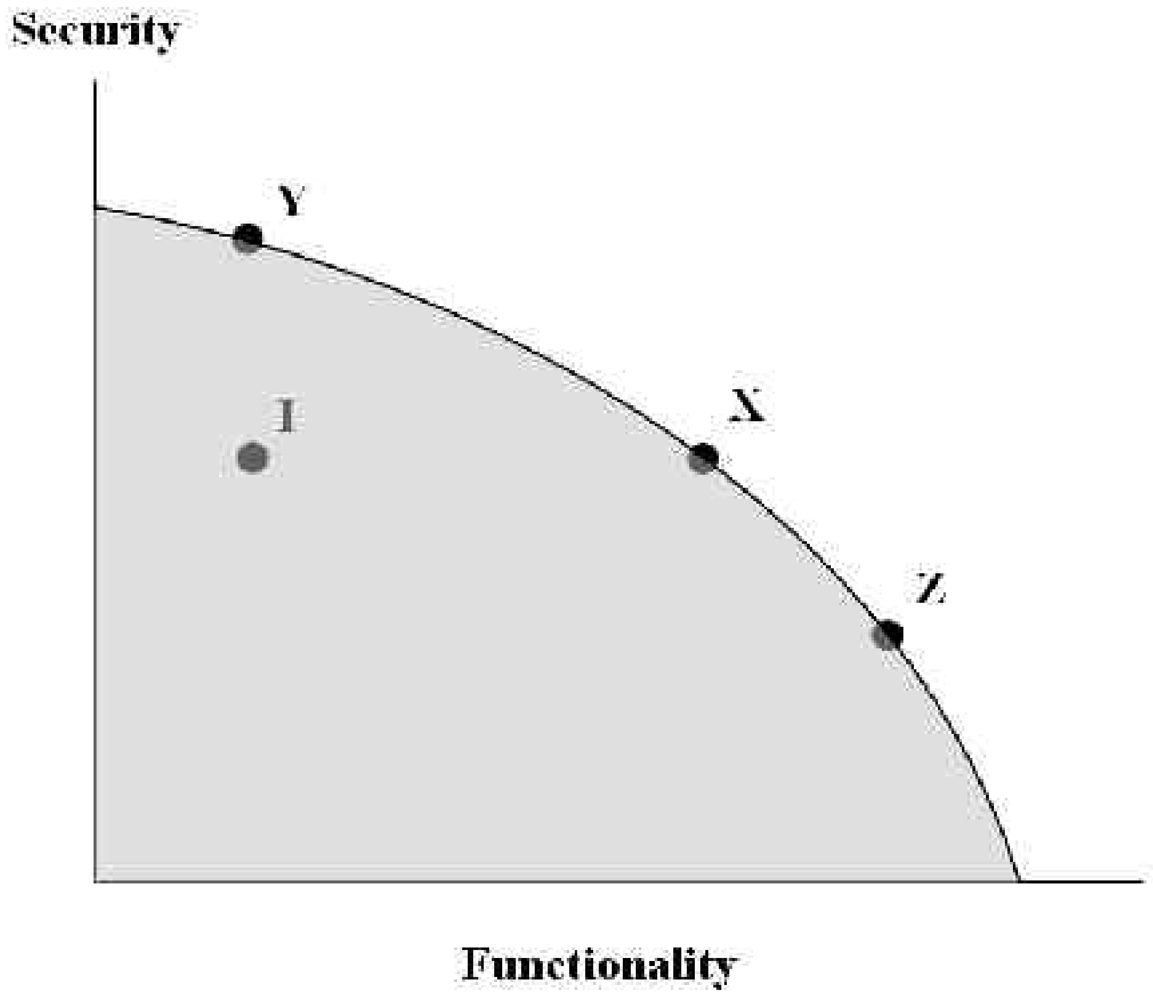

Figure 1 A simple model of the design space. Points Y, X, and Z are on the PPF. Point I is accessible but suboptimal. Points beyond the PPF are not accessible.

The key feature of our model is that there are limits to what is possible given a set of resources. If a project is operating optimally given current resources there are limits to how much security can be achieved for a given amount of functionality and vice versa. Of course, less functionality does not, by itself, mean greater security. A suboptimal point in the design space, such as I in figure 1, has less functionality but the same amount of security as a point on the PPF such as X.

Technical advances push back the PPF expanding the set of accessible points in the design space. As shown in Figure 2, this provides opportunities to increase both functionality and security. But because the basic the trade-off between security and functionality is remains, security will not necessarily be increased. There has been an enormous increase in computing power in the past 15 years. It would be trivial to develop systems which were vastly more secure than those produced a decade ago but which were just as functional. But the security of computer systems is worse now than a decade ago despite numerous technical advances in information security and computer science in general. Thus, the capacity for increased security through technical advancement does not mean that security will increase. In particular, hardware improvements have led to larger, more feature-fill, and arguably more bloated programs rather than more secure programs.

Figure 2 illustrates this trend. Here, a technical advance has moved the PPF from Figure 1 further out. X represents the status quo before the technical advance. After the technical advance, moving to point W would provide the same functionality but increased security. Instead, the trend has been to move to point D or point E, which provide increased functionality but the same security or less security respectively. Buffer overflow vulnerabilities offer a security specific example. These vulnerabilities could, to some extent, be mitigated with slower languages that perform run time bounds checking, or with tools such as stack guard [Cowan99]. But these options have not been widely utilized.

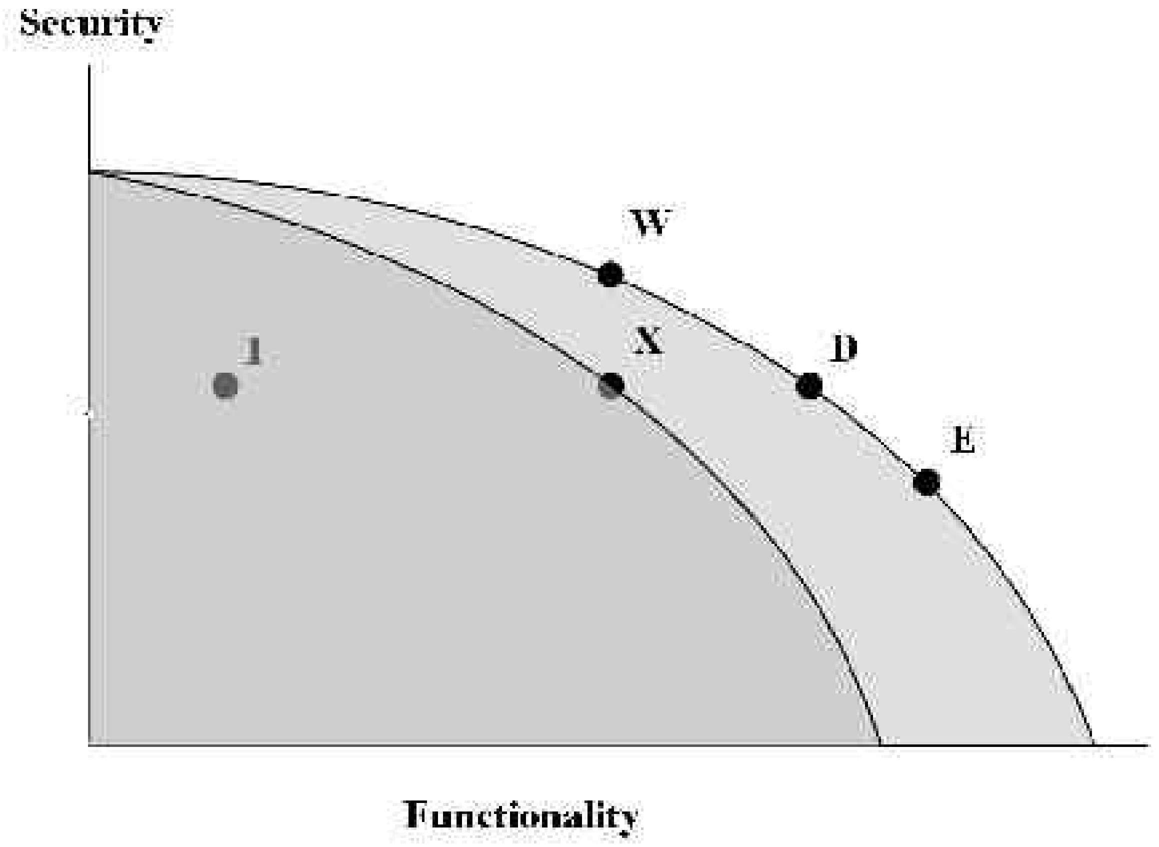

Figure 2 A technical advance extends the PPF. X is on the old PPF. W, D, and E are on the new PPF. W provides greater security than X, D provides the same security but greater functionality, and E provides less security but still more functionality.

3. Types of Technical Advances

In the previous section, we observed that the ability to provide increased security while maintaining current functionality is not sufficient to result in increased security in the market place. In this section, we attempt to explain this apparent discrepancy. Technical advances generally reduce the resources needed to obtain a particular level of security. However, if resources can now also be applied more effectively to increase functionality, the amount of functionality that must be given up to obtain a certain level of security may not be reduced. Thus the cost of a certain level of security can be best viewed as the amount of functionality that must be given up to achieve that level of security. Therefore the appropriate measure of a technical advance's effect on security is the resulting change in the marginal cost of security. In this section, we classify technical advances based on their effect on the marginal cost of security. We identify three types of technical advances: security neutral, security enhancing, and security hostile.

3.1 Security Neutral Technical Advances

Security neutral technical advances provide equal opportunities for increased security and increased functionality, and do not alter the marginal cost of security. These advances are unlikely to have a significant positive or negative impact on security.

3.2 Security Hostile Technical Advances

Security hostile technical advances increase the marginal cost of security by providing greater opportunity for increased functionality than increased security. Because more functionality must now be sacrificed to obtain a given level of security, security is likely to decrease. This type of technical advance is surprisingly common. A significant amount of research provides for increased functionality without addressing security. Especially in academia, security agnostic researchers may focus on increasing functionality and leave the resulting security issues as “future work” for more security conscious researchers to clean up after them.

3.3 Security Enhancing Technical Advances

Security enhancing advances lower the marginal cost of security by providing greater opportunity for increased security than for increased functionality. This is the implicit goal of a significant amount of security research. In the security literature, there are frequent references to the importance of making security convenient. See, for example, [Cox02].

3.4 Offsetting Advances

Software engineering has been described as a “race between programmers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots” [Cook]. Because security enhancing advances and security hostile advances have off setting effects, security engineering could be described as a race between security engineers striving to create security enhancing advances and security agnostic researchers unconcerned about sacrificing security for functionality.

The marginal cost of security will only decrease if the effects of security enhancing advances out weight the effects of security hostile advances. Promoting research in security enhancing technologies could result in increased security. However, it may be more effective to work to mitigate security hostile advances. This could be done by educational efforts to promote the importance of security and establish the view that security is an integral part of a system not after thought to be handled for others.

It is surprisingly difficult to develop security enhancing technical advances. Often, even advances in information security do not turn out to be security enhancing advances. Advances in information security often result in both increased security exposure and increased functionality in the marketplace. For example, the creation of encryption protocols that allow data to be sent over the Internet securely may have caused data to be more vulnerable because it is now more likely to reside on network accessible machines.[1] This phenomenon is by no means unique to information security. Adams argues that people have acceptable levels of risk and will compensate for technical advancements and government mandates that reduce risk by taking riskier behavior. For example, the davy lantern, which had a bulb that burned cooler than the combustion point of methane, was developed to make mining safer. Instead, the device facilitated mining in methane rich environments resulting in increased production and fatalities [Adams99]. Perrow makes a similar point in a discussion of maritime shipping. Technical advances did not create a safer industry but merely allowed greater risks to be taken such as faster travel in more dangerous weather [Perrow99].

4. Interaction with Consumer Preferences

The efficient level of security has been debated [Anderson 02, Schneier02]. Lack of security resulting purely from consumer preferences is not necessarily a bad thing. Consumers have an insatiable desire for functionality and they are willing to accept a certain amount of insecurity.[2] Indeed, efforts to reduce risk personal risk below an acceptable level can have perverse effects [Adams99].

Still there is reason to believe that increased security is socially desirable. The existence of market inefficiencies such as externalities and imperfect information suggests that the present level of security is inefficient [Anderson01]. For example, in a networked environment, insecurity in one system imposes costs on third party systems. Additionally, though the risks of cyber terrorism have been debated [Lemos02], after September 11th, governments have placed increasing importance on information security as a national security concern[3]. We do not attempt to address the policy question of whether the present

1 This is not necessarily a bad thing as functionality increases.

2 Individual aspects of a program such as speed are subject to diminishing marginal utility. However, functionality includes many properties and as defined in our model does not result in diminishing marginal utility.

3 Regardless of the actual danger of cyber-terrorism, increased stigmatization of insecure computers may be changing preferences towards more secure systems.

level of security is efficient. In this section, we examine consumer preferences and argue that the current lack of security is not entirely due to consumer apathy.

Consumer preferences strongly influence software developers, but consumer choice is often limited. The software industry is not a perfectly competitive market [Shapiro99]. There are a limited number of vendors within a particular product category. Vendors rarely offer multiple versions of a program that provide significantly differing levels of security. Auditing a program and shipping a version with known security flaws and a more expensive version in which these flaws have been fixed would raise legal and ethical issues. Reducing the size of a program by removing code that implements unneeded features can increase security because there is less code to potentially introduce security vulnerabilities. But, it is difficult to do this in practice. It is necessary to ensure that removing code does not break other parts of the program. Additionally, it may be necessary to support features once they become part of a standard or file format. Thus, consumers must select among products representing the few points in the design space that vendors have chosen to implement rather than choosing a product that provides their preferred amount of security and functionality.

However, users do not choose software purely because of its features and security. Network externalities are also significant [Shapiro99]. The utility of many products increases with the number of users. A typical user may need a certain set of features and beyond that be concerned with standardization and interoperability. In a market dominated by a single product, users who, in autarchy, would be willing to sacrifice functionality for security may choose a less secure product because of the benefits of interoperability.

In an attempt to exploit positive feed back, vendors may add features to attract a small number of users with special needs even if those features will be unused by other users and result in lower security. If consumers possess imperfect information about product security, vendors have a greater incentive to do this. [4]

5. Related Work

Our protection possibilities frontier model was inspired by the concept of the production possibilities frontier -- a standard macroeconomic model in which a country can produce

[4] It is also possible that the reverse is true. That is network effects may cause consumers to accept programs that provide more security but less functionality than they would otherwise want. However, vendors' desire to increase market share by adding additional features and anecdotal evidence that market leaders often neglect security lead us to believe that security is under provided not over provided.

two goods and faces a trade-off between the amount of one good that can be produced and the amount of the other good that can be produced. On the microeconomic level we found a somewhat analogous model in [Shapiro99]. In this model, designers face a trade-off between performance and backwards compatibility.

6. Future Work

The model described above is characterized by significant developer choice. Security and functionality can be freely traded. This is only realistic when a product is in the initial design stage. Many products have long life times of evolutionary growth and multiple releases. A complete rewrite of a legacy product is rarely practical. For example, it has been reported that Microsoft has tried to rewrite Word on multiple occasions and gave up each time [Brandy98].

A possible long term model might view the lifetime development of a product as a series of releases in which the design space for each release is highly influenced by the state of the code after the previous release. This view has interesting security implications. Security is not a feature or an add-on property [Schneier99]. If security is not designed into a system, adding it later is difficult and expensive [Hoo01]. Thus, the PPF for a project building off existing code is extremely dependent on the architecture and design of the existing code. If the existing code base emphasizes functionality over security, the PPF will be restrictive. Increasing the security of existing code requires significant expenditure of resources, which will be unavailable for increasing functionality. (As discussed in section 4, restricting the functionality of an existing system is unlikely to significantly increase security unless the restrictions are sweeping.) By contrast, if the existing code was designed with security in mind, the PPF will allow greater flexibility. Because additional resources are expended for the new release, it will be possible to add functionality while maintaining the current level of security.

Long term design becomes a strategic process. While each individual release must address immediate concerns, it also influences the possibilities for future releases. Instead of only focusing on the immediate requirements for the current design, it is necessary to look at a release's impact on future options. For example, in order to obtain increased security in a future release, it may be necessary to build the current release with more security and less functionality than its immediate requirements call for.[5]

A fully developed long term design model may well yield interesting results. For example, one possibility is that if the cost of increasing the security of legacy products is very high, then, in the long run, a product initially designed for security may have nearly

[5]Options theory may be a useful tool for analyzing these type of long term design trade-offs.

the same degree of functionality as a product not initially designed for security but will have a far greater degree of security. Furthermore, if high security later becomes a requirement, the product initially developed with security in mind may, in the long run, have a higher level of functionality if significant resources would have otherwise been expended to improve the security of existing code that are now available to add functionality. Thus a program may be burdened throughout its life by the myopic decisions of its initial developers.

We view this type of long term development model as an important area for further research. The full development of this model along with realistic empirically based estimates of the software development cost structure will allow a more thorough understanding of security in evolutionary software development processes.

7. Conclusion

The costs of security are primarily opportunity costs such as less functionality. Technical advances allow improvements in all areas, but do not change the nature of the basic trade-off. Consequently, despite significant technical advancements, security has not significantly improved though functionality has greatly increased. Technical advances that lower the marginal cost of security are most likely to result in improved security. However, often advances in information security result in increased functionality, rather than increased security. Additionally, because of network effects, users, who would otherwise prefer increased security to increased functionality, may choose less secure but more widely used programs.

8. Acknowledgments

We are grateful for insightful comments from Ross Anderson, David Feldman, Richard McGrath and Vivek Sharma. Additionally, we would like to thank Cliff Woolley and Ben Cummings for invaluable technical assistance.

9. Bibliography

[Adams99] John Adams, Cars Cholera, and Cows The Management of Risk and Uncertainty, March 4, 1999. Cato Policy Analysis No. 335, http://www.cato.org/pubs/pas/pa-335es.html.
[Anderson01] Ross Anderson, Why Information Security is Hard - An Economic Perspective, 17th Annual Computer Security Applications Conference, December 2001.
[Anderson02] Ross Anderson, 'Maybe we spend too much?', Workshop of Economics

and Information Security, University of California, Berkeley, May 16-17, 2002.
[Brady98] Robert M. Brady, Ross J. Anderson, and Robin C. Ball, Murphy's law, the fitness of evolving species, and the limits of software reliability.
[Cook] Rich Cook, as quoted in *The Quotations Page*, http://www.quotationspage.com/quotes/Rich_Cook/
[Cowan99] Crispin Cowan, Steve Beattie, Ryan Finnin Day, Calton Pu, Perry Wagle and Erik Walthinsen. *Protecting Systems from Stack Smashing Attacks with StackGuard,* Linux Expo. May 1999.
[Cox02] Russ Cox, Eric Grosse, Rob Pike, Dave Presotto and Sean Quinlan, Security in Plan 9, Proceedings of the 11th USENIX Security Symposium, San Francisco, 2002.
[Hoo01] Kevin Soo Hoo, Andrew W. Sudbury and Andrew R. Jaquith, Tangible ROI through Secure Software Engineering, Secure Business Quarterly, Vol 1 Issue 2, Fourth Quarter 2001.
[Lemos02] Robert Lemos, John Borland, Lisa Bowman and Sandeep Junnarkar, E-terrorism: Digital myth or true threat, Cnet News, August 26, 2002, http://news.com.com/2009-1001-954728.html.
[Perrow99] Charles Perrow, Normal Accidents, Princeton University Press, 1999.
[Shapiro99] Carl Shapiro and Hal R. Varian, Information Rules: A Strategic Guide to the Network Economy, Harvard Business School Press, Boston, Massachusetts, 1999.
[Schneier99] Bruce Schneier, Why Computers are Insecure, Cypto-Gram Newsletter, November 15, 1999, http://www.counterpane.com/crypto-gram-9911.html.
[Schneier02] Bruce Schneier, 'No, we don't spend enough!', Workshop of Economics and Information Security, University of California, Berkeley, May 16-17, 2002.
[Viega02] John Viega and Gary McGraw, Building Secure Software, Addison-Wesley, 2002.