Responsibility for the Harm and Risk of Security Flaws

Who is responsible for the risk and harm of software security is controversial (Dark, 2011). Deliberation of the responsibility for harm and risk due to software security flaws requires considering how incentives (and disincentives) and network effects shape the practices of vendors and adopters, and the consequent effects on the state of software security. (Dark, 2011). Unlike most engineering problems, determining degrees of moral responsibility is an ill-defined problem. Context data will always be incomplete but at the same time context data is crucial to reach a conclusion. Ill-defined problems are messy and composed of multiple problems. The solution to the problem affects one another. It is not one problem we face, but many entangled problems.

Software as we know it is changing. In the early days of personal computers, software was viewed as a product, much like a chair or a washing machine. Distinct from furniture and appliances, the increase in marginal cost to produce another unit of software is negligible. To prevent piracy, commercial software used registration keys and other mechanism that cripple software's. Today, the software industry is abandoning the product model in favor of the Software as a Service (SaaS) model. The new metaphor suggests software should be used and charged much like any service industry. One can contract software as long as he or she pays. Traditional service providers, such as a dentist, who have limited time availability, can only serve one person at a time. With SaaS software, multiple users can benefit from the very same service in parallel.

Because the mental models we use to understand the world are not static, there is an ongoing obligation to revisit the frameworks. Verifying that our beliefs are aligned with reality is common practice in the software quality assurance field. For example, the intent of regression testing is to verify that changes made to a program did not introduce abnormal behavior (aka bugs) to a program.

We turn now to discussion of the software assurance milieu. We look at the players; their roles, and their practices; incentives and disincentives; and externalities in an effort to shed light on the complexity of responsibility for harm and risk of software security flaws.

Software Manufacturers

As the creators of software applications, vendors generally bear accountability for the safety and functionality of their products. Developing computer software is a complex task. There is no rigorous way to prove that the software will even do what it's supposed to do. Examining all possible execution path combinations is neither possible nor pragmatic. A new model to review

and build software is needed. Without one, decision-makers are bound to over-invest in activities that don't generate the desired financial return, or under-invest, risking dreadful consequences.

Development lifecycle processes have been enriched to accommodate effective defense techniques that measurably reduce the number and severity of software vulnerabilities. These include secure coding techniques, minimizing the use of unsafe functions, use of compiler and linker security options, and the use of static analysis tools. Activities such as Threat modeling, security code review, penetration testing, and vulnerability management are becoming commonplace among leading software development companies (Simpson, 2008).

The introduction of additional process to the software development lifecycle requires further investment on development activities. If the goal of a company is to avoid attacks, all that is needed is to invest enough to be a less attractive target than competitors. This is like running from a bear: there is no need to be the fastest runner in the group, as long as you run faster than others, you are safe. Companies have competing investment opportunities and limited resources. "Risk" for a startup means to make next month's payroll (Chess, 2010). For many small players, security is not an afterthought: it is a conscious decision to defer investments to a later time when value of digital assets surpasses a perceived threshold (Anderson, 2001).

Software Adopters

Users buy new products because of features. To make new features you need more developers. To make more reliable features you need quality assurance personnel. There is an implicit trade between security and features users are willing to accept. When customers buy feature rich software without worrying about security, they incentivize the trade of security for features by voting with their money. Customers may do this because they are not educated to judge whether a product is secure. From a security perspective all products look equal. Some companies may be willing to do this trade to fulfill customer desire as customers won't pay more for the difference they cannot see. The market will then drive the price of "invisible features" down. To stay competitive, responsible vendors need to drop best practices to comparable levels practiced by the competition.

Software adoption in enterprises is a decision influenced by several factors other than technicalities. Companies are willing to forgo the best technical solution if discounts or strategic partnership opportunities are bundled in a deal. Often, management simply chooses market leader products believing this negates the need for proper due diligence. Because choices are

related to economics and company strategy, the decision makers are not the same people who will use and support the software. There is a disconnect between analysis and operations.

For example, today's biggest IT expenditures are software, followed by hosting solutions (i.e., hardware) and network infrastructure. When it comes to security, the budget structure is reverse (Grossman, 2010).

Software vulnerabilities often come from environment misconfiguration and poor operations (Greenemeier, 2007). Some of the biggest computer security breaches in recent history originated from these two vectors of attack. TJ Max's credit card leak occurred due to improper wireless network configuration. Google's source code leak in China relied on insider information.

Security researchers

On the premise that meaningful, high quality vulnerability research should be rewarded and that customers have a right to know, firsthand, about vulnerabilities that may impact revenues, companies such as iDefense and TippingPoint have emerged (Dark, 2011). These companies purchase vulnerabilities from finders. Once a vulnerability is tested and confirmed by these companies, they notify their subscribers and appropriate software manufacturers about the finding. iDefense and Tipping point do not go public with the information and neither do any of the customers. Instead, the companies assume the finder's duty of communicating with the vendor and provide sufficient information to create the fix (Dark, 2011). Once the fix is out, iDefense and TippingPoint pay the finder for the research. Some note that this phenomenon, the creation of a market to profit from software flaws, seems ethically questionable (Dark, 2011).

The sale of information about a single security flaws in a widely deployed product is valued up to US$40,000, the equivalent of a full year worth of salary in developing countries. The same vulnerability market that rewards researchers for finding vulnerabilities attracts disgruntled employees to purposely create security flaws and later indirectly sell the flaws through an agent.

It is in a software manufacturer's best interest to look after the secrecy of unfixed vulnerability information to protect both customers and their own reputation. The same due care cannot be expected from companies that buy vulnerabilities to sell protection services: information leakage boosts subscription value and increases the number of subscribers. Finally, the true intent of a subscriber is not known. The same information that is used to protect against attacks can be reverse engineered to attack non-subscribers.

Government

Ever since people began to live together in society, laws have been necessary to hold that society together. Laws help to ensure a safe and peaceful society in which people's rights are respected. Although a retail store can take reasonable precautions to reduce the risk against robbery, no defense is flawless. To some extent, stores rely on laws to protect assets that will always be at risk. If, likewise, software makers exercise reasonable due care building computer systems, it is the government's responsibility to enact laws to further protect users and try to ensure that no one will exploit any remaining flaws left on the system.

Enacting effective laws on evolving technologies may result in unforeseen side effects. For example, one way to contain the spreading of computer viruses is to cut internet access of unpatched computers disseminating the plague. While this sounds like a straightforward approach, network content monitoring privileges must be granted to an organization to implement the system – a dangerous tradeoff between security and privacy. Further, because small software manufactures tend to have more vulnerabilities in their code than mature software manufacturers, laws must not facilitate monopolies by stifling new entrants and innovation.

Conclusion

The rate of innovation in technology continues to be faster than any other discipline and the dynamics of the software industry and characteristics of computational devices will certainly change. Improving software security is as much about economics, public policy, and social welfare as it is about abuse cases, error conditions, and testing methodologies.

Acknowledgement

This paper is based on chapter 6 of Information Assurance and Security Ethics in Complex Systems: Interdisciplinary Perspectives authored by Cassio Goldschmidt, Melissa Jane Dark and Hina Chaudhry

References

Anderson, R. (2001). Why information security is hard - An economic perspective. Retrieved from http://doi.ieeecomputersociety.org/10.1109/ACSAC.2001.991552

Chess, B. (2010). : Watching Software Run: Software Security Beyond Defect Elimination. Presentation at OWASP FROC 2010 Conference. Boulder, CO. Retrieved From http://blip.tv/file/3710067

Dark, M. (2011). Information Assurance and Security Ethics in Complex Systems: Interdisciplinary Perspectives. Hershey PA: IGI Global

Greenemeier, L. (2007, May). T.J. Maxx data theft likely due to wireless 'wardriving'.

Information Week. Retrieved from http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=199500574

Grossman, J. (2010). Facilitating Application Security Maturity. Presentation at OWASP AppSec Brasil 2010.

Simpson, S. (2008, October). Fundamental Practices for Secure Software Development – A guide to the most effective Secure Development Practices in Use Today. Retrieved from http://www.safecode.org/publications/SAFECode_Dev_Practices1108.pdf

Recommended Reading

Anderson, R., & Moore, T. (2006). The economics of information security. Science, 314, 611.doi:10.1126/science.1130992

Elf, D. (2004). [Full−Disclosure] iDefense: solution or problem? Derkeiler. Retrieved May 7,

2009 from http://www.derkeiler.com/pdf/Mailing-Lists/Full-Disclosure/2004-07/0698.pdf

Rice, D. (2007, December). Geekonomics: The Real Cost of Insecure Software. Addison-Wesley Professional

Schneier, B. (2007, January). Debating full disclosure. Schneier blog site. Retrieved from http://

www.schneier.com/blog/archives/2007/01/debating_full_d.html

Schneier, B. (2008). Software Makers Should Take Responsibility. Retrieved September 27, 2009

from http://www.schneier.com/essay-228.html