ACM: Digital Library: Computers and Society

Computers and Society

Volume 34, Issue 3 (December 2004), Page 1

Economies of disclosure

Jeff Bollinger

Table of Contents

Introduction

Imagine this scenario: a bank customer walks up to an ATM to withdraw cash from her account. While entering her PIN, she accidentally presses the '3' key at the same time as the 'Clear' key. Instantly $100 comes out of the cash dispenser! Curious, she checks the receipt and seeing that the money did not from her account, she tries the same operation. Again, $100 comes out of the cash dispenser. At this point she has two options, A: she can continue to take advantage of this obvious flaw and tell her

friends, or B: she can report this problem to the bank, telling no one else. While Kantian ethics might require the latter, the issue becomes much grayer when it comes to computer software vulnerabilities. Many variables exist to cloud the ethical judgment of a software flaw's discoverer. The question becomes: is it better to report any vulnerability that could cause catastrophic problems (as in the ATM example) or to make the vulnerability information public in order to simultaneously compel the software vendor to address the problem while giving the vendor's customers a chance to prepare for potential exploitation? Software companies are facing this dilemma on a regular basis. There are dozens of websites, newsgroups, and e-mail lists dedicated to the task of sharing vulnerability information, whether condoned by the software company or not. The issue increases in severity as vulnerable software becomes ubiquitous across global networks. The public release of vulnerability information is often the first drop of the monsoon that is the malicious network worm. How this information is controlled and disseminated is critical to the stable operation of millions of computer systems and networks across the globe.

Modern software with network capability is fundamentally complex and can span thousands of lines of source code for one small part of the package. Without proper testing and quality assurance, many bugs and errors slip through. Software development requires these iterations of coding and bug fixing to produce stable and polished final products. Larger and more complex software gives a greater chance for a bug to remain undiscovered. Yet, since major vulnerabilities have appeared in relatively smaller projects, proper code audit and testing are critical regardless of the size of a software package or the amount of source code required to produce it. If bugs or security vulnerabilities are not discovered and fixed by the software manufacturers, it is inevitable that they will be discovered by outside researchers.

This breeds the ethical conundrum of what the researchers do with the

information once it has been discovered. Foundational to disclosure, regardless of the ideology, is the notion of providing useful protective information to the most appropriate audience. Many suggest that disclosure is about doing the 'greatest good' for the greatest number of people, which is an obvious adherence to the ethical principle of Utilitarianism. There are numerous parties involved in the network security process such as software vendor, security researcher, system administrator, and even malicious attacker: all whose intentions reflect on the ethical principles behind their actions.

Responsible Disclosure

There are numerous theories and practices for disclosing information about a software vulnerability. Many advocate the notion of 'Responsible Disclosure' as a means to inform software vendors of major problems (Rasch, 2002). Responsible Disclosure advocates call for an intermediary between the developers and the researchers who can relay information between the two parties and ensure the issue is resolved. With an established disclosure process, software companies have the ability to track and prioritize a report based on their evaluation of the findings and the necessity to release a patch or other workaround if necessary. Major issues with Responsible Disclosure surface when the process breaks down. Researchers accuse vendors of both poor quality control in their software design and failure to respond to their findings, while the vendors must take the time to evaluate each bug submission for its validity and potential impact. Rasch also suggests that, ' no vendor wants to publicly admit that a product was released with a vulnerability ' (Rasch, 2002).

Responsible Disclosure does not address the fact that malicious attackers often know information about a security vulnerability well before the software company or security researcher. Having this knowledge of an 'undiscovered' vulnerability is a major power shift for the attackers. Given that these types of vulnerabilities often lead to '0-day' attacks, or attacks

which occur without any knowledge or speculation as to their origin; the malicious attackers have a substantial advantage over unpatched computer systems. This makes it critical to have accurate vulnerability information available to quickly prepare for any attacks. Jeremy Rauch argues that Full Disclosure does not necessarily mean that vendors do not have the opportunity to respond to bug submissions. He writes,

'Contrary to popular belief, vendor notification and full disclosure are not mutually exclusive. Many people choose to notify vendors prior to disclosing information and give details to the public only after the vendor has had an opportunity to address the problem. Since the real goal is to improve security, it is rare for people to post a vulnerability without giving thought to its impact' (Rauch, 1999).

Regardless of the ideology for determining the propriety of a disclosure process, the reality is that attacks will still occur and there is a potential for damages when any exploit code has been released. Preston and Lofton propose that, 'publication of code that circumvents any kind of computer security on the Internet will result in people circumventing that security' (Preston and Lofton, 2002). They question the supposed good intentions of releasing dangerous code, since the discloser knows it is likely the code will eventually be used for nefarious purposes. Without specific measurement and reassuring data that security threats are actually reduced by a specific disclosure, it is philosophically inconstant to subscribe to the Utilitarian approach with the undeniable uncertainty of the net benefits. Yet the detraction from the inherent Utilitarian approach within the Full Disclosure ideology is mitigated given the notion that any disclosure is better than a secretive release, or a total lack of disclosure. Arbaugh, Fithen, and McHugh suggest in their 'Vulnerability Life-Cycle Model' that,

'When someone discovers that a product has security or survivability implications, the flaw becomes a vulnerability. It doesn't matter if the

discoverer's intentions are malicious (Black Hat) or benign (White Hat). It is clear that, in many cases, original discovery is not an event that can be known; the discoverer may never disclose his finding' (Arbaugh, Fithen, and McHugh, 2000).

The act of disclosure in itself is beneficial to the security community and substantially less dangerous than a vulnerability that is never known. An unprepared and thus, compromised system would devastate administrators who could not have adequately prepared their systems.

Full Disclosure

In contrast to Responsible Disclosure and in response to the secrecy of vulnerability discovery and the associated detractors, the practice of 'Full Disclosure' has come into purview within the security community. According to SecurityFocus co-founder, Jeremy Rauch, Full Disclosure involves 'disseminating information about security vulnerabilities. It is not about any one aspect; it is not publishing specific vulnerability exploits, nor is it about embarrassing vendors. Its sole purpose is to arm the security-conscious with the knowledge necessary to evaluate risks and take applicable action' (Rauch, 1999). Information about a vulnerability is announced publicly and typically in an open discussion forum where the ramifications and potential consequences can be debated. This methodology for disclosing information has come under severe scrutiny from software vendors for the obvious reason that they are forced to either patch their software, or announce their position on a particular vulnerability as soon as the disclosure has been released. This forcing of the hand by the security researchers is an ethically questionable attempt to compel the software company to react rapidly on numerous fronts, including: public relations, customer relations, patch development and deployment, and resolving interoperability issues. It can put the company in a tense situation, yet the researchers would argue that this situation is self-designed, and if the company had either prevented the vulnerability

in their quality assurance process or had responded to prior notifications of the bug, they would not be required to respond at all. Rauch also argues that disclosure and open dialogs on security problems actually increase the security of software. He writes,

'buffer overruns were once an obscure topic, but now they have been discussed and dissected to the point where many programmers understand how to prevent them, even if they are incapable of writing an exploit. Where there were once ten new exploits based on the buffer-overrun concept each week, the rate at which they are found has slowed to a trickle. The discussion of these problems in an open, collaborative forum helped to promote understanding, which in turn has significantly reduced the number of vulnerabilities of this type'In the same way that an open-source project benefits from the input of programmers worldwide scrutinizing its code base, system security has benefited from the scrutiny of full disclosure.' (Rauch, 1999)

Researchers might also contend that a vendor may simply choose to quietly ignore a vulnerability submission. Large software companies rely on the capability and pervasiveness of their software and an announcement of a major flaw in its design can be dangerous to their business and customer relations. In many cases, the companies have the resources to dedicate to bug detection and removal, yet they continue to release seemingly unchecked code.

Do software companies have an ethical obligation to actively respond to a potential vulnerability report, regardless of who submits it? Scott Culp, the program manager for Microsoft's Security Response Center writes in a 2001 essay that,

'At the end of the day, a vendor's paramount responsibility is to its customers, not to a self-described security community. If openly addressing vulnerabilities inevitably leads to those vulnerabilities being exploited, vendors will have no choice but to find other ways to protect

their customers' (Culp, 2001).

The key to determine whose actions are ethical in the disclosure debate boils down to the intentions and methods of the discloser, and the response of the software company. Following each one Smith and Hasnas's (1999) 'Normative Theories of Business Ethics', a software company should respond to complaints about potential vulnerabilities to protect the stockholders, stakeholders, and in some cases to be socially responsible. High-profile or very prolific software packages or operating systems are especially important to protect, given the ease of which software vulnerabilities can 'morph' into fast spreading worms. Not responding appropriately or timely to a major vulnerability report can be disastrous for the public relations of a software company if they are forced to admit publicly to the shortcomings in the software that thousands of people and businesses rely on. This violates all of the Normative Theories and lowers the trust and authority that customers place in the companies. Software vendors must ask themselves, what is our obligation to our customers to provide information on potentially dangerous software flaws and their associated fixes, and to what extent are we willing to go to protect their customers and products? Regardless of prior expectations of complicated licensing agreements, customers will hold the vendor responsible for major problems with their product whether a vulnerability was discovered by the company or not, and set a reasonable expectation that the vendor will respond appropriately to any potentially damaging issues.

In addition to ethical concerns with failure to act on a report of a potential vulnerability, there may also be legal issues surrounding the failure to repair or to close major vulnerabilities before they appear publicly. Nancy Wahl indicates that its possible for a software vendor to be held liable for a vulnerable product that directly causes a loss. She writes,

'ordinary negligence applies when a software developer does not use the

degree of care that a reasonably prudent person would have used when developing software. For example, negligence could include the following: failing to adequately design, code and test a program; neglecting to warn of consequences of improper use; and not explaining how to install a program correctly' (Wahl, 1994).

Failure to code and test a program constitutes a reasonable definition of negligence, and it may be possible for a customer to argue that a vulnerability in a software package that was exploited on their network caused hours of down time and extra efforts on the part of IT staff to repair the problems. However, software vendors can and do skirt this legal trouble by disclaiming these types of problems and often provide the software 'as is'. A plaintiff would have to prove that the software licensing agreement contained language that led them to believe the software manufacturer would be responsible for vulnerabilities or potential damages as a result of their negligence. Wahl suggests that, 'the best solution to the problems caused by unreliable software is to create only reliable software,' and that 'to avoid financial loss because of defective software, the developer must identify the risk and understand the extent of the risk. Employers are responsible for negligence committed by software developers. Therefore, it is in the best interest of the employer to hire qualified people, supervise them carefully, and provide continuing education' (Wahl 1994).

Wahl contends that prevention through training and careful development is the best method to avoid vulnerable software. It follows that the prevention of egregious software errors is in the best interests of the company and that they are ethically obligated to ensure that they are taking the appropriate measures to at least meet the expectations of their paying customers. The customers expect and pay for usable and secure software. The software vendor, then, is responsible for meeting the demand and not misleading the customer, while the researcher needs to appropriately address the vulnerability and give the vendor time to

research them. The researcher must also be diligent in establishing their intent on the submission of a vulnerability announcement as Preston and Lofton indicate when summarizing the judgment from *Brandenburg v. Ohio* , 'the context of speech, specifically the speaker's intended audience, determines its criminality' (Preston and Lofton, 2002).

The Exposition

Central to this controversy are the security researchers. Whether full-time professional, non-profit, or amateur, the number of security and vulnerability researchers is large enough so that dozens of vulnerabilities are announced everyday in various forums like the Bugtraq, Vuln-Dev, and Full-Disclosure mailing lists. While the ethical concerns of many researchers are nebulous at best and criminal at worst, there seems to be a consensus on these mailing lists that it is important for everyone to be informed from a public forum about the potential problems and risks involved in running certain software. The ethical nature of the discloser is firmly rooted in their intentions when they release a vulnerability report after a discovery. Since it is nearly impossible to accurately gauge a discloser's intention on the release of a report, it is difficult to determine the ethical ramifications of such a release. It is possible, however, to speculate the possibilities for intent, though such speculations can never quite justify any damages that may be caused by a particularly grave disclosure. Jay Heiser suggests that most 'researchers' are nothing more than fame seekers who are out to discover vulnerabilities in software for recognition. In his column in *Information Security* from January 2001, Jay writes that,

'Having discovered that they can attract huge amounts of attention by throwing rocks at [Microsoft ©] Windows, so-called security professionals are increasingly the ones fulfilling both the research and the application stages. Sadly, the shortest path to computer security fame seems to lie more in providing candy to children than in breakthroughs in dental

hygiene. The concept of full disclosure is, indeed, ambiguous, serving as a politically correct shield behind which all manner of self-serving behavior can be justified. It's far too often used to rationalize shortsighted information releases that benefit the announcer to the detriment of the entire Internet community.' (Heiser, 2001)

This argument proposes that the intent behind a vulnerability release is primarily negative and that it violates any Utilitarian ideologies, even if the outward intent is to protect system administrators from system compromise. He continues hinting at Utilitarianism when he compares the choice between non-disclosure and full-disclosure: 'Instead of striving for the optimum compromise, just choose one of the poles. You won't have to think hard, and you have the added satisfaction of moral superiority' we need to ask what activities will produce the greatest common good.' (Heiser, 2001)

The notion of 'greatest common good' strikes at the heart of the issue of disclosure, with each stakeholder in the discovery and publication process having a different interpretation of the maxim. Large software corporations are unlikely to advocate a public announcement of a security flaw in their products and may even suggest that releasing public information about a vulnerability does much more harm than good. Ethan Preston and John Lofton write in the *Whittier Law Review* , Richard Smith, the director of the Privacy Foundation, criticized eEye for releasing information about the vulnerability in IIS, arguing that eEye's publications indirectly caused the release of Code Red.' (Preston and Lofton, 2002) The Code Red worm of 2001 saw a major global attack of Microsoft web servers as a result of a vulnerability in their server software. News.com writer Robert Lemos indicates that: 'In June, eEye found the security vulnerability in Microsoft's Internet Information Server that is being used by the worm. Known as the index-server flaw, the security hole was detailed and patched by Microsoft more than a month ago.' (Lemos, 2001)The software companies and other opponents argue the intentions

of the researchers do not reflect the optimal methodology for releasing security information, leaving unprepared administrators at risk for a major attack. The notion that public disclosure of software vulnerabilities leads to massive destruction via Internet worms or viruses is as critical an argument against the advocates of Full-Disclosure as the conception that system administrators do not effectively patch their systems. Scott Culp dubs Full-Disclosure as 'Information Anarchy' and writes,

'Supporters of information anarchy claim that publishing full details on exploiting vulnerabilities actually helps security, by giving system administrators information on how to protect their systems, demonstrating the need for them to take action, and bringing pressure on software vendors to address the vulnerabilities. These may be their intentions, but in practice information anarchy is antithetical to all three goals. Providing a recipe for exploiting a vulnerability doesn't aid administrators in protecting their networks. In the vast majority of cases, the only way to protect against a security vulnerability is to apply a fix that changes the system behavior and eliminates the vulnerability; in other cases, systems can be protected through administrative procedures. Likewise, if information anarchy is intended to spur users into defending their systems, the worms themselves conclusively show that it fails to do this. Long before the worms were built, vendors had delivered security patches that eliminated the vulnerabilities. In some cases, the fixes were available in multiple forms ' as much as a year in advance. Yet when these worms tore through the user community, it was clear that few people had applied these fixes.' (Culp, 2001)

Culp seems to suggest that the onus of responsibility for keeping systems secure belongs to system administrators, or those who use software regularly and have a stake in its proper operation and maintenance. Culp and others argue that software is increasingly more complex and that vulnerabilities and security holes will continue to exist, perhaps more so, given the higher complexity requirements for future releases. He is

simultaneously arguing that the cause of destruction and gross system down time are symptomatic of the failure of administrators to apply security patches and fixes in a timely manner, and that security researchers who recklessly publish vulnerability and exploit information are contributing to the 'anarchy' of the security process. Preston and Lofton simplify Culp's argument by writing, 'Culp argued that ethical culpability lies with computer security publishers that enable attacks rather than the vendors of insecure products and users who fail to diligently apply patches' (Preston and Lofton, 2002). They also suggest that Culp argues, 'it is better to suppress security issues and weaken demand for security than to increase the demand for security by calling attention to security issues.' (Preston and Lofton, 2002) This implies that disclosure in any format is a negative sum for everyone involved in the process. Culp criticizes information disclosures and links them directly to exploits and attacks. His arguments against disclosure not only suggest that researchers are ethically responsible for any damages, but also contest the theory that disclosure yields a greater good than suppressed 'security issues'.

System Administration and the Security Patch

Arbaugh, Fithen, and McHugh suggest that the act of disclosure is not what prompts widespread damages, but that the 'scripting' or automation of attack methods to affect a broader range of victims is the true source of significant damage. They indicate,

'In our research, we found that automating a vulnerability, not just disclosing it, serves as the catalyst for widespread intrusions. In each case study, patches for the vulnerability were available at least a month before target sites reported intrusions to CERT. Further, the patches were generally available shortly after or concurrent with the vulnerability's public disclosure. Thus, while open disclosure obviously works, the availability of patches prior to the upswing in intrusions implies that

deployment of corrections is woefully inadequate.' (Arbaugh, Fithen, and McHugh, 2000)

This research proposes that the parties most responsible for widespread damage from security intrusions are the malicious attackers who automate exploits from previously disclosed vulnerabilities, and system administrators who do not properly install security patches or perform regular security maintenance. Given an obvious affiliation and potential contribution to the spread of a dangerous worm, what ethical role do the system administrators play in the security process? Since it follows that imprudent patching schedules directly contribute to the spread of vulnerability exploits, perhaps administrators are behaving unethically in their lack of action to prevent exploitation and the subsequent spread of attacks. Arbaugh, Fithen, and McHugh discovered that,

'the most compelling conclusion from this research, however, is the surprisingly poor state in which administrators maintain systems. Many systems remain vulnerable to security flaws months or even years after corrections become available'We did not, however, anticipate that almost all reported intrusions could have been prevented had the systems been actively managed, with all security-relevant corrections installed' (Arbaugh, Fithen, and McHugh, 2000).

Administrators may be partially at fault for neglecting their systems and furthering the spread of Internet worms. Just as tenants would be indirectly responsible for the damage if they failed to prevent a fire in their apartment building and it spread to other rooms, a system administrator is comparably obliged to exercise the due care that is required to prevent system intrusion and the resulting down time and potential data loss or damage. They have an obligation to their users, networks, and business partners similar to the obligation that the software vendors have to them. Furthermore, with respect to ethical operations, an administrator should not neglect to properly protect a system from an exploiting worm to affect

the operations of other systems on the Internet. They need to understand the ramifications of an attack, and its plausibility on their systems and networks. Therefore, administrators need public disclosure information, with particular regards to a vulnerability's effect and mitigating factors, in order to properly balance the usability and availability of their systems to its security requirements.

Power of Information

Having intimate knowledge of a security vulnerability gives administrators the knowledge to properly defend their systems from related exploits. The ethical question remains: how should that valuable information be disseminated, if ever? Jeremy Rauch asks, 'There is always potential for an exploit that is made public to cause major damage. The question that's difficult to answer, however, is: What would the impact of the problem have been had it not been disclosed' (Rauch, 1999)? Vendors, system administrators, and security researchers all have a stake in the security process. Their intentions and behaviors throughout the lifecycle of a vulnerability determine their adherence to any ethical principles. Since Utilitarianism is inherent to the advertised general mantra of these researchers, they must ensure that their intent and their audience are properly considered before putting the security of the Internet's system administrators and the reputation of software vendors at risk. The individual intent measures the adherence to an ethical ideology after a disclosure. Intent can be presumed by looking at the target audience, the intended message, and the behavior exercised in preparation for and during the release. The evident breakdown and source of controversy for the Full Disclosure debate surfaces from its inability to provide acceptable accountability for every party involved in the security process. It does not consistently work in a Utilitarian manner, often causing or inciting major problems on the Internet. To effect the optimal result of 'greatest good', each player in the disclosure process must agree and co-ordinate to achieve the greatest return, and lowest damages. The due care exercised

by the researchers to promote knowledgeable security information as opposed to spreading fear and uncertainty, in addition to the care exercised by administrators to apply the requisite patches serves as ethical behavior that effectively extracts the greatest good for the largest percentage of people.

References

1. Arbaugh, W. A., Fithen, W. L, and McHugh, J. (2000) Windows of vulnerability: A case study analysis. *IEEE Computer* . 33: 52-9.

2. Culp, S. (2001) It's Time to end information anarchy. *Microsoft.com: Security Essays.* Available online at http://www.microsoft.com/technet/columns/security/essays/noarch.asp Accessed on December 14, 2003

3. Heiser, J. (2001) Exposing infosecurity hype, Full disclosure? Full complicity! Deconstructing the myths behind the full-disclosure debate. *Information Security* . (2001, January). Available online at: http://infosecuritymag.techtarget.com/articles/january01/columns_curmudgeons_corner.shtml Accessed on October 13, 2003.

4. Lemos, R. (2001) Code Red worm set to flood Internet. News.com. available online at: http://news.com.com/2100-1001_3-270263.html . Accessed on December 14, 2003.

5. Preston, E. and Lofton, J. (2002) Computer security publications: information economics, shifting liability and the first amendment . *Whittier Law Review* . 24: 71-142. available online at: http://www.mcandl.com/computer-security.pdf

6. Rasch, M. (2002) 'Responsible disclosure' draft could have legal muscle. *SecurityFocus* available online at: http://www.securityfocus.com/columnists/66 Accessed on December 12,

2003.

7. Rauch, J. (1999). Full disclosure: The future of vulnerability disclosure? *;login: The Magazine of USENIX and SAGE*. 11. available online at: http://www.usenix.org/publications/login/1999-11/features/disclosure.html

8. Smith, H. J., and Hasnas, J. (1999) Ethics and information systems: the corporate domain. *MIS Quarterly* . 23(1): 109-27.

9. Wahl, N. J. (1994) Responsibility for unreliable software. Ethics in the computer age. Proceedings of the Conference on Ethics in the Computer Age, 175-7.

Authors

Jeff Bollinger, CISSP, University of North Carolina, IT Security Analyst, 105 Abernethy Hall, jeff@unc.edu