

**Ethical Repercussions of Public Security Vulnerability Disclosure**  
by Jeff Wurz and John Roehrig

2014 Words

When one discovers a programming error in a piece of software that compromises its integrity, they are faced with an ethical dilemma. Choosing to release the vulnerability to the public may empower malicious individuals to exploit running instances of the software. Keeping the vulnerability private and documenting it when a corrected version is released has associated risks as well. Never releasing information of the vulnerability may cause users of the software to continue using the vulnerable version, not seeing any need to upgrade. Of course, some also choose to keep the vulnerability to themselves so they may exploit it until someone more just discovers the error.

The simplest programming error can yield catastrophic security vulnerabilities. Lacking a simple bounds check while setting a variable which is dependent on some action of the user can open up the possibility for this user to acquire the privileges of the user running the vulnerable program. Worse yet, a program operating over a socket would allow a remote user to break into a secure network. This can be done with BIND, a TCP application publicly accessible on most networks providing Internet services which had many denial of service and remote code execution vulnerabilities earlier this decade and in the late nineties. These vulnerabilities are incredibly well documented; people with very little programming ability are able to exploit these errors to their own advantages. Even worse are the countless CGI vulnerabilities discovered around the time of the Internet bubble. Workforce shortages caused companies to hire unskilled programmers and pressure them to design server-side applications in a hurry to take advantage of the seemingly endless supply of venture capital up for grabs. This resulted in a downturn of programming quality creating a flow of vulnerable code being published on various web servers. Common programming errors led to discovery of these vulnerabilities without observation of the code.

Though most people see hackers as a threat to banks and large companies, their destructive potential reaches over the breadth of society. Identity theft at the hands of computer criminals sometimes affects people who never even use computers. Let us consider someone who hacks a television home shopping channel's network and acquires an explicit list of customer information. It is very likely that many of this channel's customers are without access to the Internet, yet they are greatly affected by it. When I was younger, my friends and I used to gain access to public security cameras. From this simple public security breach, one can easily see how this may affect someone with no association to the Internet whatsoever.

Obviously the exploitation of security vulnerabilities for personal gain at the expense of others is an unjust act in itself. In most developed countries, it is considered a serious malum in se crime. Unfortunately it would seem that the motivation to discover vulnerabilities is primarily held by the malicious hacker. These hackers only need to find one vulnerability to successfully hack into a system. On the other hand, the server administrator understands that these vulnerabilities exist, and he or she discovers them without the intent to do harm

to others' systems. When an ethical person discovers a vulnerability, he or she is faced with a complex decision, wherein all options carry a distinct ethical repercussion.

Vulnerabilities are often found when private source code is scrutinized by company employees for closed-source software. For many years, companies would simply fix the vulnerability and release the patched software in a later version, keeping the existence of the bug a secret. Users who were not motivated to pay for the newer version of the software usually remained vulnerable and unaware of this fact. What if these vulnerabilities were not in software, but in an automobile? When defects are discovered that compromise the safety of a car, governments usually forcibly compel the manufacturer to recall the car and right the problem at no cost to the consumer. Does this same standard apply to other companies when they discover a product to be unsafe? Do they have a moral obligation to correct the problem? I believe they do have such an obligation. Perhaps proportionately to the increase in popularity of open-source software, companies have been increasingly disclosing vulnerabilities and offering free fixes. Some applications even have complex systems implemented to automatically update vulnerable software quickly and effortlessly making knowledge of these vulnerabilities even less important to those who automatically update by their own software.

Publicly disclosing these vulnerabilities makes the public aware of their existence. Not only does this empower the benign consumer to protect himself from a malicious hacker, it also empowers those same hackers to do their evil deeds. As we all know, detecting vulnerabilities in closed-source software is much more difficult than scrutinizing open-source software's code for unchecked bounds and common coding mistakes. Does quickly disclosing the existence of these vulnerabilities create an opportunity for blackhats to formulate a method of exploitation and wreak destruction on systems that have not been made invulnerable? Some companies choose to release patches to software describing vague descriptions of the vulnerability so as to compel the user to update the software without disclosing information to would-be hackers.

Perhaps the nature of closed-source software actually provides some protection against vulnerabilities. Without empirically observing code, it is very difficult to detect vulnerabilities.

However, this is not true when one considers WinNuke, an infamous denial-of-service attack that exploits an error-checking vulnerability in older versions of Microsoft Windows. These older versions panicked when following a pointer from an OOB packet that pointed to the end of the frame. This bug existed for years before it was discovered, and when it was, many versions of Windows were found to contain the vulnerable code. Millions of people were exposed and many an IRC-pup and script-kiddy was empowered by this newfound the ability to crash a target's machine with the click of his or her mouse.

Though closed-source applications by nature have more vulnerabilities due to the fewer number of people scrutinizing the code, usually only those working on the code are aware of these vulnerabilities. When bugs are discovered less frequently, software does not need to be updated as frequently. Bugs like the one WinNuke exploited are a rare occurrence. Usually when these vulnerabilities are discovered, patched, and not disclosed to the public, there is never a chance for it to be exploited. Legacy software will eventually fade from existence, with its usage becoming too infrequent for people to care about exploiting. None will ever feel harm from these bugs, so as long as no one discovers them outside of the developers bound by confidentiality agreements. Though these "WinNuke" bugs are rare, they are quite detrimental. Perhaps WinNuke is what provoked Microsoft to develop automatic Windows Update instead of relying on infrequent Service Pack updates.

The consequences of disclosure or nondisclosure of vulnerabilities in open-source software are very different from those in the close-source realm. As any subscriber to BugTraq will tell you, the number of known vulnerabilities in open-source software that are discovered every month is staggering. It is, after all, the nature of open-source software to be heavily scrutinized by many people. But what should one do when heavily scrutinizing this code and one discovers it is vulnerable to malicious exploitation? There are many courses of action: contact the developer(s), discuss the vulnerability on Usenet or some other open discussion, fix the bug and distribute a patch, etc. One course of action, unarguably unjust, would be to simply ignore the error; or worse, use this exclusive knowledge for personal gain at the expense of vulnerable hosts.

Vulnerability forums around the Internet are probably frequented by more malicious users than ethical ones. As previously explained, early discovery of vulnerabilities is more beneficial to the hacker than to the system administrator. While the hacker can compromise many hosts before the vulnerability becomes well known and recognized, the system administrator can only make his network marginally more secure. Though the act of publishing this information may empower the hacker more than the system administrator, the public community should have a right to know if the software they are using is vulnerable without relying on the discretion of the development team to disclose the information. Many people choose a form of deceptive disclosure. They disclose the vulnerability publicly but deliberately publish an error in the exploitation information. Perhaps these people assume that if one is intelligent enough to see through the deception, then one is also an ethical person. This is an obvious fallacy. To avoid disclosing vulnerabilities to malicious hackers some people may choose to skip the online forums all together and submit their findings directly to the development team.

In the same realm of thinking, there are also many security vulnerabilities in open source software as well as closed source. When considering the affect that releasing these bugs has on those using the software, it may be concluded that the overwhelming majority of those actually working on open source code are not looking for security holes. Many of these people are finding out how specific functionalities are asserted and using this information in some

other code they might be writing. They would only pay attention to that which would help them achieve their goal. There is also a small number of closed-source programmers who are given the sole task of looking for security flaws in software that has open-source code. One such company, Google, encourages and has programmers just for that specific purpose. By using some of their massive resources, they are paving the way to a large base of very secure code.

The method of software updating for open-source software on Linux is very easy and can be automated. There are large software repositories which one can be searched at set intervals of time. When new patches and updates of software one has installed become available, the user is notified and asks for the update to be downloaded. As soon as this task is completed those security holes will be filled and one can consider the system safe again.

The open-source realm of software does take longer than the closed-source software to identify vulnerabilities. This is only because there is so few people looking at the code, and even from those not all are looking for flaws. If a large group of eyes were to start systematically checking all the open-source code for security bugs, a massive increase in software security would be achieved. It has been said by Eric Raymond, “with many eyeballs, all bugs are shallow.” From this maxim it can be concluded that the more people who are looking through a piece of code, the better the chances are of flaws being found.

Keeping our software up to date and bug free should be every computer user’s goal. There is no giant force of programmers fixing our software for us. We need to do it for ourselves. This open source movement should bring all computer users closer to the codes we execute every day. With understanding of code comes a greater ability for average computer users to find better ways to have safe software.

As one can see, there are many things that must be considered when one discovers vulnerabilities in software. The course of action one chooses to take when one discovers a vulnerability can have severe consequences on people who use the software. This course of action may empower malicious individuals or rob users of their right to use secure software. When disclosing vulnerabilities an inherent conflict exists between the rights of the users of the software’s right to protect themselves from intrusion and the users of the software’s right to be protected from intrusion. We don’t beleive that there is any particular right or just action to take after discovering vulnerabilities, though there are definitely unjust courses of action. What one ought to do is beyond the scope of this paper and is beyond what these authors are willing to conjecture.