Computer Science Professional Ethics Final

Mitchell Dodson - April 24, 2021

Discussion 1 - protecting personal data

In this scenario, the subject professional is responsible for creating a networked system for distributing critical private information of patients of a social services clinic among several machines, some of which need to be stored on devices such as laptops and phones that will frequently be outside of a physically secure premise. The security of information stored on these devices is an extremely high priority since a compromise in the system resulting in the leakage of patient data could enable the distribution of highly personal information about the most vulnerable parts of patients' private lives, and could even physically endanger patient safety. The clinic contracting the software has a limited budget, so clinicians may mistakenly underestimate the risk and be willing to make undue concessions because they aren't aware of the gravity of the situation, or may not consider the security of the system in the first place because it is out of their field.

Stakeholders in this scenario include the clinicians who task the professional developer in creating the system, the patients themselves, policymakers and sources of private funding for the clinic (such as donors and investors), and the professionals who participate in programming the system. The most important and vulnerable of these are the patients, who assume no responsibility in determining the system's architecture, are likely legally guaranteed the privacy of their records by medical standards, and who stand to lose the most if the system is faulty. Furthermore, directors and clinicians who use the system are pinning their own professional integrity on the stability and security of the program, and implicitly trust that the developers will take appropriate measures in building a safe system. Finally, policymakers, sources of funding for the clinic, and donors are similarly staking their personal and professional integrity on the security of the system, and failures to preserve security may result in funding being taken away from a clinic otherwise providing important social services, disproportional mistrust of systems like the one being developed, or policy changes that could have more broad negative consequences.

There are a few important considerations to keep in mind while determining the most professional and ethical course of action. First, it's important to know the type and sensitivity of information that will be stored on the system. Clinicians likely have their own professional guidelines on handling the specific types of information that need to be stored, and by ACM code 2.6 (which states that ethical professionals should only perform work within their realm of competence) the specifics of handling patient information should first conform to the clinic’s standards. Additionally, the clinic’s ignorance of security risks mean that an ethical developer should take it upon themselves to effectively communicate vulnerabilities and their implications to the contract holder, taking care not to over-estimate or under-estimate their likelihood and the consequences of a breach.

According to *A Gift of Fire*, “Most of the people affected by … devices, systems, and services of professionals do not understand how they work and cannot easily judge their quality and safety, (which) creates responsibilities for the professional.” (p405). Therefore, the dominant difference between the way the non-professional “everyman” and a professional software developer should be expected to handle an issue like this one boils down to the awareness of potential security risks, and the initiative to find and implement sufficient security measures to handle the risk. As stated above, the burden of discovering and handling technical risks of a networked system like this must fall on the developers since the problems are outside of the clinician’s field of knowledge. An everyman wouldn’t be expected to have an awareness of the issues, much less to know how to effectively mitigate them, and as such a professional is uniquely responsible for handling these concerns effectively and efficiently.

Ultimately, I believe the best initial course of action for the professional developers is to make a valiant effort to clearly and completely explain the risks to the contract holders as well as potential ways to mitigate risks (and the costs/benefits of these methods) in a way they can fully understand. If the clinicians are sufficiently cognizant of the problems at hand and the options you have for solving them, they can make informed decisions on the system they want with respect to the standards put forward in their own field and the constraints of the budget. If the contract holders still insist on risky decisions due to obstinance, lack of resources, or any other reason, an ethical professional still has a responsibility towards the most vulnerable of the stakeholders - the patients. In this case, the developers should could insist on clearly disclaiming the risks to the patients so they can at least make an informed decision on whether to use the system. Even still, due to the sensitive nature of the patients’ circumstance there may not always be another option for the service. As a last resort, the developers could refuse to implement the insecure system - which may effectively convey the magnitude of the situation to the clinic - or could consult higher management or sources of funding for the clinic, who may be able to provide more funding for sufficient security and who likely have more influence on the clinic’s decisionmaking than the contractor. On a more technical level, there are probably cheap or free solutions to most of the security issues such as encryption or more stringent authentication. The professional should fall back on the best possible solution given the constraints.

Discussion 2 – Professional Responsibilities

Computer scientists and programmers occupy a privileged role in society in that their work is seen as esoteric to the broader population, but is imperative to the function of everyday life to everyone. Whether or not people realize it, they depend heavily on the diligent work of programmers for everything from their privacy and personal security to the effectiveness and efficiency of work in their own fields of expertise. The ABET requirements for undergraduate degree programs recognizes this with the stipulations that graduates must be able to communicate effectively in a professional context and that graduates must be able to recognize their responsibilities as programmers and make decisions based on legal and ethical principles.

ABET's requirement that programmers have "the ability to communicate effectively in a variety of professional contexts" takes into account the fact that many colleagues who aren't professional programmers may have a poor understanding of the process of software development, the implications/risks of some decisions in the creation of a product, and may not have the same intuition as programmers on the nature of software products they need to use. Addressing the first point, an example of a common frustration of many programmers is pushback from management or customers due to difficulty in meeting time or budget limitations for products. Requirements for software products are typically rather abstract, based on the function and behavior of a final product instead of the details of implementation; most professionals who aren't computer scientists don't have the desire or ability to easily understand the specifics of program architecture, and may be dissatisfied by what seems like slow progress. The "everyman" would be tempted to try to explain similar shortcomings from their own perspective, may consider the criticism unfair or personal, and may try to explain the issues exhaustively as the "everyman" sees them, but an effective professional computer programmer will realize that the lack of knowledge and intuition of those raising issues validates their concerns. The programmer should take into consideration the customer's desires and understanding of the program and formulate an impartial and clear explanation that addresses their concerns and remains honest without overwhelming them with technical jargon.

Second, effective communication is critical when colleagues who aren't programmers are unable to fully evaluate implications and risks of a program due to their lack of technical understanding. ACM code 2.5 posits that professional programmers should "Give comprehensive and thorough evaluation of computer systems and their impacts." This is fundamentally important because – whether or not they realize it – the average non-programmer relies on software developers to make important decisions on issues that directly affect them. In a professional context, managers and customers have to trust that programmers will inform them of the potential effects of implementing software requirements. Similarly, software products must be paired with documentation and training that sufficiently articulate to users the proper ways to interact with them. In *A Gift of Fire* page 407, the author includes an anecdote about a catastrophic opening day for a new airport. The project manager blamed the result on the airline clerks' failure to use the software designed to run the airport's logistics, which the text claims to be fallacious. If end-users are unable to effectively use otherwise-functioning software to accomplish its goal due to ineffective training/documentation, then the software has not accomplished its goal after all; given the esoteric nature of software design and implementation, it's up to the developers to ensure that the product can be entirely utilized by its users.

According to the ABET guidelines, software developers also need to take due responsibility and make decisions based on legal and ethical standards. One context for which such principles are especially relevant to professional programmers is in the preservation of intellectual property rights. Information stored on computers including programs, text, media, et cetera are inherently transient, easy to find, and easy to distribute, which makes the integrity of intellectual property difficult to maintain.

As with the privacy, free speech, and innumerable other issues, the onus falls on programmers to protect intellectual property in the domain of software they create. According to *A Gift of Ice and Fire* (page 183), the estimated cost of pirated software alone is valued in billions of dollars. This issue is famously faced by services like YouTube, which has been criticized by its heavyhanded approach to the protection of intellectual property rights. The YouTube content claiming system gives alleged victims of IP theft the right to any monetization of stolen content as well as the right to demand that the content be removed from the platform entirely. Although for the specific example of YouTube's content policy there are additional problems created by inefficient validation of ownership and faulty preemptive claims by their neural network-based content evaluation system (which raises its own ethical concerns), strong methods of preventing IP theft and piracy may be required to uphold creators' rights in an online world where information flows like water. Professional programmers must take into careful consideration the ways in which their software contains or interacts with intellectual property and make every effort to protect it from theft and misuse by unethical actors.