The risk of using proprietary software

Do you know what you're feeding your computer?

Matt Barton

About one out of every 200 people is allergic to peanuts. Depending on the extremity of the allergy, a person suffering from peanut allergies who was accidentally exposed to peanuts might develop an itchy rash. Others might experience anaphylaxis, a severe reaction that can prove fatal. People who are allergic to peanuts have a tough time in America, where more and more foods are manufactured in factories that also process peanuts.

Thankfully, manufacturers and restaurants are coming under pressure to clearly label foods that either contain peanuts or were prepared with machinery that also processed peanuts. These measures have saved lives and helped Americans live healthier lives, because knowing what you're eating ought to be important to everyone. Without access to this information, hundreds of men, women, and children would die each year. People with a deadly sensitivity to peanuts would literally be playing a game of Russian roulette every time they tried a new food.

What if we lived in a nation where manufacturers weren't required to print the ingredients of their foods on their packages? What if food corporations had successfully lobbied Congress to allow them to keep their ingredients totally secret? We can easily imagine the arguments. "If we publish our ingredients, then our competitors will be able to duplicate our recipes". "We use chemicals and ingredients that might turn off consumers". "If people don't trust us, they shouldn't eat our food; we shouldn't be *forced* to list the ingredients". Nevertheless, in this case, common sense won out, and now we're entitled to not only read the basic ingredients but also get fairly reliable nutritional information about the foods we eat. We don't necessarily consider that the manufacturers are graciously offering us a service. Instead, we see this as our *right*, a demand that we reasonably make to manufacturers. If a manufacturer refused to tell us what is in its food, we would be stupid to eat it anyway. It's just *common sense* to require that manufacturers tell us what they're putting into our foods - because we put those foods *into our bodies.*

You have a *right* to know what's going into your food, you have a *right* to know what a piece of software is doing inside your computer

Now, let's consider another case that isn't really much different than food, but is nevertheless treated as though it *were*: computer software. No, I'm not saying that you should try munching your copy of *Half-Life 2*. What I mean is that software is something that you put in another type of body; namely, your personal computer. For the same reason that you have a *right* to know what's going into your food, you have a *right* to know what a piece of software is doing inside your computer. It ought to be *common sense* that software developers be required to publish this code for your review before you run their programs. When a software developer tells you, "No, just *trust us*," your mental

Fig. 1: Never has a game been so aptly named

red-alert should start sounding loud and clear. This is a dead giveaway that you should steer clear of this software and not even consider installing it on your machine. Sure, perhaps it is safe and legitimate. But how do you know? Is it worth taking a risk with all of your precious programs and data? Why should you even be asked to take this risk?

The propriety of trust

Let's explore this concept a bit. Let's say you are browsing the games at your local software shop and find a great new role-playing game from a major developer. Even though it's a bit pricey at $60, you're impressed with the description on the box and take it home. Unfortunately, after playing the game for a few hours, you decide you don't really like it. It's boring and not nearly as good as you thought it was going to be. Of course you can't take it back to the store since no one is going to trust you enough to believe you didn't make an illegal copy of the game. So, disgusted, you decide to use the game's uninstallation program to take this clunker off your hard drive.

This program deletes the entire contents of your hard drive. Gigabytes worth of papers, emails, family photos, and countless other valuable data is lost forever.

Yeah, right, you say. This would never happen. Yet it did.

The game is Stormfront Studio's *Pool of Radiance II: The Ruins of Myth Drannor,* distributed by Ubisoft and released in 2001 (see this IGN review (`http://pc.ign.com/articles/162/162043p1.html`) or this Game Over.net review (`http://www.game-over.net/reviews.php?id=663\&page=reviews`)). If you haven't heard of it, there's good reason. The game was one of the most pointless and sleep-inducing games since *E.T.* for the Atari 2600. In the world of big-budget commercial games, this isn't really anything unusual. It's also expected that there will be plenty of bugs, some of them show-stopping bugs, in early releases that will only be fixed later on by downloadable "patches" and "fixes." However, the development team responsible for *Pool of Radiance II* represents an all new low for proprietary development: The uninstallation script can actually damage vital system files and has reportedly wiped some users' hard drives completely.

Of course, the developers soon released a patch to replace the dangerous uninstall program, but is that enough to help us sleep better at night after installing a new proprietary program on our computer?

I started this article by describing why we, as a society, demand that food manufacturers tell us what they put in our food. What would be the equivalent practice we should demand of software developers? The answer is that they should release all of their source code so that we get a chance to *see* what their programs will do to our machines *before* we install and run them.

"Wait a minute," you say. "I don't know *anything* about software code. I'm *totally* code illiterate. How is that supposed to help me? I wouldn't be able to tell what the software was doing to my computer even if I had the source code!"

Well, you *could* learn to code. It's not impossible, and, in fact, not really more difficult than learning how to read or learning to speak another language. I'm of the opinion that everyone should learn at least the basics of programming. The rewards are immediate and immense.

But, let's say that you don't care and will *never* care about knowing how to program. Why would having access to the source code matter to you?

The answer is that while *you* may not understand the source code, there are plenty of other people who do. These people would be very likely to spot malevolent, dangerous, or just outright sloppy code coming from the developer and publish

their discoveries on the internet for all to see. If you read that a new game contained code that could delete random files on your hard drive, you'd know better than to install it. There are plenty of people out there who would happily perform this public service, and they would get something out of it, too. By having access to the code, they'd be able to learn quicker and faster how other programmers are working their magic.

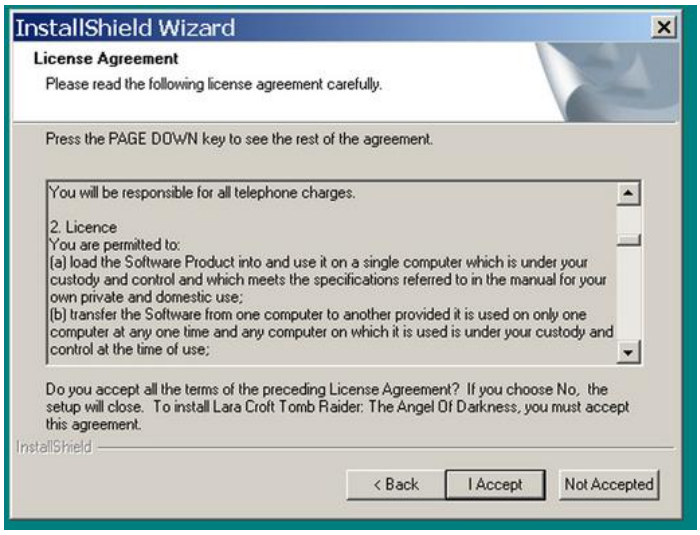
In short, proprietary developers depend on your naiveté - your ignorance and gullibility - to install and use their software. They say "Trust us," and that's supposed to be sufficient. If you don't like it, you don't have to buy their products. Now, keep in mind that the proprietary developers expect you to trust them with all of the programs and data on your computer - but do they trust you?

Quite the contrary. In fact, most commercial developers have taken it upon themselves to include programs that prevent you from making a legitimate backup copy of the software products you purchase. They've also made your life more difficult by asking you to enter long and complicated registration or verification codes into your software before it will install on your drive. Many games require that you find and keep the CD-ROM inside your drive anytime you want to play it. Other copy protection schemes manipulate your system so badly they may trigger your anti-virus protection programs! There are even reports that some copy protection programs scan your drive and will not allow a program to be installed unless you remove certain programs first. Should I mention other "features" like region-encoding? Do you enjoy having to click "I ACCEPT" to pages of incomprehensible legalese before installing a product you just paid $60 or more to own?

No, the proprietary software industry doesn't trust us. In fact, it is so suspicious that it is willing to compromise the efficiency and convenience of its programs in a vain attempt to thwart hackers from making and distributing illegal copies. They don't even consider you the "owner" of the software you purchased; it's just "licensed" to you. If you don't know the difference, don't worry - the developer's attorneys will be happy to explain it to you in court.

Let's put things in perspective. When you buy a proprietary program, you're expected to just have faith that it won't damage or destroy your computer. On the other hand, the makers of the program don't trust you one bit. In fact, they take every possible precaution to restrict your freedom and

Fig. 2: A typical end-user agreement. No, you won't be negotiating the terms of this "agreement"



guarantee that you'll obey the rules they establish to regulate your behavior.

Does this sound like a good deal to you? Of course not. But is there any alternative? After all, we have to have software to run on our computers if we want to get any work done or have any fun with them.

The free software alternative

The good news is that there is a wonderful alternative to proprietary software. You guessed it: free software, and it's becoming more plentiful and effective every day. Now, don't get the wrong idea: free software isn't necessarily free as in it doesn't cost you anything. Folks who work long and tedious hours developing great software have a right to ask for compensation and often do. When people talk about "free software," they're not describing software in the public domain, which *is* free and doesn't cost anything. Free software is simply software that has been released under a *public license*, such as the General Public License (or the GPL). If this is confusing, just think about what we mean when we say "free speech." This doesn't mean that you go the library and start stealing books. It means that somebody can't use the government to force you to shut up or not print something just because they disagree with your opinion. You're free to use it, copy it, study it, and improve it however you

see fit - you don't have to answer to anybody, even the people that developed and released the program.

Let's put this in context. Assume that you want to use a software program that will make your screen flash "Microsoft sucks!" over and over again. Now, if this screensaver program is proprietary, the owner of that software can tell you that your use for that program isn't appropriate, and represents a violation of your "End User Agreement". In fact, if you don't stop, you're going to be sued. You aren't free to use this program in a way that the developer doesn't approve of. It's a *non-free program*. In fact, instead of just seeing "I Agree" when presented with end-user agreements, I think you should see "Yes, Master". This terminology would much better reflect the type of relationship you are entering when you install a proprietary software program. On the other hand, if your screensaver is a free software program, it doesn't matter what the developer says. If she happens to see you running the program and calls to ask you to change your message, you can tell her where to shove it. However, I doubt very seriously such a preposterous thing would ever happen, because free software developers would be the first people to tell you that you have a right to say whatever you want.

Free software also has another huge advantage: the source code is always available for your review. If you want, you can even compile it yourself so that you know *for sure* that it isn't doing anything questionable to your data. If you decide that you don't like the way it operates, you can either fix it or find someone who can. Of course, you may have to pay this person to make the changes you need, but it's an option that you don't have if you're using proprietary software.

Conclusion

Some people like to downplay the importance of having free software. They say, "Well, if the program does what you need it do, it doesn't matter if it's free or proprietary". Bill Gates has taken to referring to free software developers as communists, hell-bent on undermining democracy. Such critics always manage to make their position sound like pure common sense, even though what they're preaching is the opposite of common sense. No sane person would eat whatever was handed him without bothering to find out what it was. No sane person would claim that demanding that we hold manufacturers responsible for their products is "communist". Indeed, it's far more "communist" to believe that we should just let other people make these decisions for us; that it isn't really our business and that we should just trust in our leaders to do what's right.

I believe that people are becoming more aware of the dangers posed by proprietary software developers and are getting tired of playing their game. We have a right to know what we're putting into our computers, and even if we can't read code, we know that other people can and will tell us if something is seriously wrong. We're also getting fed up with developers who don't trust us enough to let us make backup copies of the programs we purchase, yet expect us to trust them with the precious data stored on our computers. Finally, we're sick of being told how to use the programs that we buy and that we're not allowed to change them if we want.

You *do* have a choice, but that choice is not likely to come from the big companies that have been making a fortune selling you proprietary software and dictating how you should use it. When you decide that you've finally had enough, then it's time to learn about GNU/Linux and the thousands of free software alternatives that are just as good (if not better) than their proprietary equivalents. If you're totally new, a good place to start is KNOPPIX or SIMPLY MEPIS. These free operating systems are easy to install and fully functional. You've tried tyranny; now see how you like freedom.

Copyright information

About the author

Matt Barton is an educator and writer who is currently living in Tampa, Florida. He is an advocate of free software and the Creative Commons. He hopes to receive his Ph.D. in May 2005.