Security and open source

Roger Needham

Security problems in software are of course an extremely bad thing, regardless of the business model under which the software was written. I want to consider why anybody thinks that the business model matters, and whether there is evidence that it does. I shall also look somewhat to the future.

Are security bugs different from ordinary bugs? This is a relevant question to follow Ross Anderson's talk. On balance I would claim that they are, not for a technical but for a social reason. Consider a paradigmatic "ordinary" bug, such as a library that wrongly calculates the square root of 2, while apparently doing everything else right. After a certain amount of hilarity the community response would be either to use a different library, or, more likely, to avoid taking the square root of 2. If a security bug is found in a system there is a community of people who make it their personal priority to make the wrong behavior happen, typically in other people's computers. This must affect any kind of statistical analysis.

Security bugs, like ordinary bugs, are often found when they happen. They may, and I believe often do, actually happen as ordinary bugs. A system misbehaves, the cause turns out to be a buffer over-run, and an experienced person realizes that the bug could have been exploited to take control of the machine – i.e. the bug gets characterized as security and receives urgent attention.

What has this got to do with the business model under which the peccant software was written? It is relevant because much has been made of alleged increases in security of OSS software by reason of scrutiny by a huge number of eyeballs (for some reason it is always quoted as eyeballs not brains). There isn't any actual evidence that this works better than what commercial vendors do. This is not to say that security bugs are never found that way, but to say that there is no evidence that it's a better way than careful professional scrutiny by people who are trained to do it. I would not make too much of the absence of evidence either way, because there isn't any very good metric for the secureness or otherwise of a piece of software. The number of distinct reported security bugs is at best a rather weak proxy for an ill-understood notion, and indeed it shows very little difference either way between software written under different models. It may be that the degree of security of a software system is like the health of a nation – governments choose measures that show that things have improved under their regime but the significance of the whole is doubtful.

In fact there is some curious relevant evidence from Microsoft’s Shared Source program. Since that started many, many more people have been in a position to scrutinize Microsoft’s programs but the security bug identifications have not been coming from there. I do not find it too difficult to see why. Many security bugs are in out-of the way and unglamorous pieces of code, and unless you’re being paid to look at it why should you. It’s most unlikely that, for example, Ross Anderson and his students, who are people well qualified to do this sort of thing, would actually spend substantial time doing so – and his career prospects would not be advanced if he did! It is one thing to look at pieces of code that have undoubted security sensitivities, such as implementations of crypto algorithms and the like. This is intellectually stimulating and generally fun – and may occasionally yield results, often of the form that there are much fewer bits of randomness in some crucial quantity than was supposed or required. It’s quite another to look at an implementation of a public standard such as TCP/IP to check that there aren’t obscure errors. This is a tedious and usually unrewarding activity.

Now a look at the future. It is becoming more and more the case that computer applications are not on standalone machines. They use services from elsewhere (Web services) and split responsibilities between the end user’s machine, the prime vendor’s machine, and the machines from which the vendor purchases services. In some real sense, the computing platform is not the PC but the Internet. This means, among other things, that the responsibility for security for the end user becomes much more diffuse. It changes the nature of the security responsibilities of platform vendors as against service providers. The purchaser of a Web service wants to know that what he buys will not be maliciously interfered with. His assurance is likely to depend on the security of the software which is run on the end user’s own machine, and of the platform vendor from whom the service supplier purchases, and on the security of the system the service supplier supplies, which is where the supplier’s added value is. The service the service supplier supplies is highly likely to depend on Web services supplied by others. The question is, what does this do to security requirements, regardless of which business model the software is produced with?

It seems to me that security will be driven by contractual relationships between suppliers and their subcontractors. If I want to offer a service to customers, and my service depends on Web services offered by others, then I may want an assurance from my subcontractors that they will deliver what I paid for. How will I get such an assurance? On the one hand the subcontractor may be a big enough concern that I can accept its assurance, and rely on contract clauses about indemnity. On the other hand I may need to take out insurance against damage to my customers through act or default of my suppliers. I would suggest that in the future the insurance industry may play an important role in deciding which software to underwrite.

One of the topics I alluded to above was the requirement that the software running in the end user's machine be appropriate. This is a topic of enormous interest and controversy. If the good behavior (= meeting the requirements of a vendor) requires that the software running in the end user's have thus and such characteristics, how is this to be assured?

There is no doubt that people interested in making sure that the software that is used to manipulate content that they own is as they wish need, or feel they need, to control the programs that are run on other people's computers. This is a wholly new dimension to computer security. In the traditional approach to the subject we have two participants, say me and the Bank of America, engaging in a transaction. I want to be assured that the other participant in indeed the B of A, and I hope the B of A has an equal interest in knowing that I am me. We both want to be assured that the transaction goes forward in accord with our mutual intentions.

In the area of requiring that software running in the user's machine to be appropriate, we enter a quite different area. Unlike the scenario between me and the B of A, where the enemy, so to speak, is the rest of the world, the scenario is that I, the owner of the computer, am the enemy. The assumption is that I wish to do something contrary to law or agreement and that I should be prevented from doing this. The vendor of a service I use does not want me to alter that part of the software that runs in my machine to cause his service to fail and me to sue him. It is a similar area of endeavour to making sure that correct (i.e. non copy generating) software be installed on PC’s – in fact to Digital Rights Management.

Whether DRM is desirable, allowable, or otherwise proper, is not my subject. Some people think it is a good thing, some people think it is a very bad or dangerous thing – but this is not the subject of this meeting. Unfortunately one aspect of DRM is pertinent to today’s subject. If the owner of a computer is regarded as the enemy then something must be kept from him. This is not a wholly novel situation. It is assumed that the owner of a highway diesel truck will, if he can, increase its performance at the expense of the emission regulations; it is assumed that someone selling a used car will, if he can, wind the odometer back. For a European audience one may also refer to tachograph problems. In all these cases some actions are deliberately made difficult even for the owner of the equipment. The consensus among security professionals I have discussed the matter with is that if you want to do DRM without hardware support – another topic that is not for today – then there is little choice but to rely on some level of secrecy or obfuscation. Similar things are true of Pay-TV. Anyone in the security field will deplore this necessity, but given the desire to achieve the results necessity it is, and it goes against openness and sharing.