7. Security and Privacy as Market Failures

Most software is sold as is, according to the end user licensing agreement. This means that the software is released with bugs, known and unknown. There is tremendous market pressure to push software out the door. Money flows in as soon as the software is written. Being first may mean market dominance.

The resulting software, with errors, may need to be repaired regularly. Indeed, one Tuesday a month for the life of Windows XP Microsoft released patches to address failures in the software. This is software that runs critical systems, and that had been on the market for years.

In part this is because the information economy is relatively new. Code is complex. Even line by line examination of code cannot detect every possible security failure. Some bugs, i.e. vulnerabilities, are a result of interaction between programs. The errors emerge in complex unforeseen ways, as the miracle substance DDT resulted in poisoned birds as well as dead bugs. Some errors emerge not for interaction, but only from unique states in the program. There are bugs that occur only when tens of conditions are simultaneously met, so that they could not be detected even in the most rigorous testing.

The result is that the individual who does not keep his or her machine secure by patching every month is at risk. However, every machine that is not patched creates risks for everyone connected to the network. Like the industrial facilities and home toxins that poisoned communities in the first half of the twentieth century; software vulnerabilities and unpatched home machines poison the network and the information on which virtual life depends.

In formal terms, lack of security can be seen as a particular kind of market failure, an externality. (Camp & Wolfram, 2001) Computer security failures cause downtime and costs to the people other than the ones who either create or do not mitigate these failures. At the most obvious, stolen information enables identity theft. But at a more subtle level, the Internet is a network of trust.

Three common ways in which lack of security on one system harms another are shared trust, increased resources, and the ability for the attacker to confuse the trail. Shared trust is a problem when a system is trusted by another, so the subversion of one machine allows the subversion of another. (For example, when passwords for one machine are kept on another). The use of cookies to save authentication information has made this practice extremely common.

The second issue, increased resources, refers to the fact that attackers can increase resources for attacks by subverting multiple machines. This is

most obviously useful in brute force attacks, for example in decryption or in a denial of service attack. Using multiple machines makes a denial of service attack easier to implement, since such attacks may depend on overwhelming the target machine.

Third, subverting multiple machines makes it difficult to trace an attack from its source. When taking a circuitous route an attacker can hide his or her tracks in the adulterated log files of multiple machines. Clearly this allows the attacker to remain hidden from law enforcement and continue to launch attacks. The last two points suggest that costs to hackers fall with the number of machines (and so the difference between the benefits of hacking and the costs increases), similar to the way in which benefits to phone users increase with the number of other phones on the network.

A fourth point is the indirect effect security breaches have on users' willingness to transact over the network. For instance, consumers may be less willing to use the Internet for e-commerce if they hear of incidents of credit card theft. This is a rational response if there is no way for consumers to distinguish security levels of different sites.

Because security is an externality the pricing of software and hardware does not reflect the possibility of and the extent of the damages from security failures associated with the item.

Externalities and public goods are often discussed as if they are the same. They are two similar categories of market failures. A common example of a public good is national security, and it might be tempting to think of the analogies between national security and computer security. National security, and public goods in general, are generally single, indivisible goods. A pure public good is something which is both non-rival (my use of it doesn't affect yours) and non-excludable (once the good is produced, it is hard to exclude people from using it).

Computer security, by comparison, is the sum of a number of individual firms' or peoples' decisions. It is important to distinguish computer security from national security (i.e. externalities from public goods) because the solutions to public goods problem and to externalities differ. The government usually handles the production of public goods, whereas there are a number of examples where simple interventions by the government have created a more efficient private market such that trades between private economic parties better reflect the presence of externalities.

Identity management systems may be a public good. For identity management systems to work they either need to be dedicated to a specific use, or usable by all. If one person can subvert an identity management system, then everyone is at risk for subversion.

SoBig, a virus hat made a splash, is an exemplar of security as an externality. SoBig was motivated by the ability to subvert the computers of naive end users in order to implement fraud through phishing and spam. The creator of SoBig has not been detected by law enforcement. In fact, the lack of consideration of agency in computer crime laws creates criminal liability