UC Davis
Computer Science

Title
Technology, Training, and Transformation

Permalink
https://escholarship.org/uc/item/9n44886v

Journal
IEEE Security and Privacy, 8(5)

Author
Bishop, Matt

Publication Date
2010-09-01

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Basic Training
Editors: Deb Frincke, Deborah.frincke@pnl.gov
Richard Ford, rford@se.fit.edu

Technology, Training and Transformation

Matt Bishop
University of California, Davis

As advances in technology transform society, so must it transform how people interact with that technology. However, there's a disconnect here: technology advances much faster than do society and people. Witness the glacial pace of change within the law. The ability to make perfect copies of digital media has existed as long as digital media has existed. Rather than changing to cope with these technological changes, current law attempts to legislate against the use of the technology—an effort that will ultimately fail, given human nature.

This raises the question of how transformations in technology affect security. Specifically, how does the ubiquity of technology, a key change in our society, also transform how we practice security? How will people and companies not in the technology realm protect themselves?

Computers vs. Cars

One solution that has gained widespread currency draws an analogy between computers, cell phones, and appliances that use computers (called "computer-enhanced devices") with widely adopted technology, such as automobiles. The argument goes that people know how to use their cars safely and "in the large," in the sense of using brakes to maintain safe distances from other cars, locking cars to prevent theft, driving defensively to reduce the likelihood of accidents, and carrying insurance to assist if any of these practices fail. Thus, the argument goes, as people have learned how to operate cars safely, so must they learn to operate their computers, cell phones, and other computer-enhanced devices safely. Owners and users must learn to secure their gadgets and, more important, take responsibility for doing so—or accept the consequences of not doing so.

The problem with this analogy is the difference in timeline, complexity, user community, and packaging. An automobile has a (relatively) simple user interface, honed by a century's use, and key components are standardized (such as the relative positions of the brake and the accelerator). Most people don't know how to repair a car when something goes wrong. They take the car to a mechanic to have it fixed. Similarly, government organizations mandate standards that automobiles must meet, and manufacturers can be penalized for selling cars that fail to meet these standards. Most countries also have controls on the use of cars, too. In the US, young children aren't legally allowed to drive. Drivers must pass both a written and driving exam, and obtain a driver's license that not only identifies the driver but also specifies any restrictions on driving (for example, needing to wear corrective lenses). There's also an existing infrastructure that's relatively easy to understand, at least in the US. People and companies aren't left adrift in making safety and security choices when it comes to their cars. Yet even with these standards and requirements for licensing drivers, there are still accidents, unsafe cars, and ways to steal or manipulate cars and their components.[1,2]

Computers today have complex interfaces, each one unique. Unlike cars, computers have been in widespread use only for about 15 years, and mobile devices even less. People often don't know when something is wrong, in part because they don't have enough experience with computers and in part because computers don't distinguish between transient errors ("unable to download attachment") and persistent ones ("computer botnet detected"). Also, some problems don't leave an easily observable footprint, such as password theft. Vendors typically don't provide assistance with software; in fact, most software warranties specify that the software is sold "as is," and any incidental damages are the user's responsibility, not the vendor's. Finally, there are no security standards for software used on most computers.

This implies that a user seeking to be both safe and secure must understand his computer far better than a driver must understand his automobile. Were the analogy to be accurate, parts of cars would be sold piecemeal, possibly assembled at home, and owners (or drivers) would be responsible for assembling the parts into a whole. They would also have to know how to test the components because they're sold "as is."

While some people do put together cars this way, it's usually considered a hobby rather than a typical household task.

An alternative is to restrict the use of computers to those who are willing and able to learn to secure them. This is analogous to requiring you to have a "driver's license" before you can use a computer. Presumably, people who want one would need to show competence in using a computer safely, as well as securing their own system (to protect not only themselves but also others). What follows focuses on the latter part.

Such an approach has two problems. The most obvious has just been described: it's unrealistic to expect poets, writers, and cooks to become computer security experts. Indeed, given that systems maintained by experts are penetrated—witness the compromise of government and corporate systems—why should we expect the home user to be able to harden his or her system enough so that it's immune to penetration and compromise? Were such a requirement in place, both corporations and governments would immediately have to stop using computers connected to the Internet—an unlikely scenario.

The second problem is more subtle. Denying any group of people the ability to use computers would require barring them from using microwave ovens, television sets, and other common home appliances. As automobiles now have many computerized devices, they too would be forbidden. Indeed, the ubiquity of computer-enhanced devices makes this alternative unenforceable.

One possibility is to have a set of computers—for example, embedded systems—that are unrestricted, and other sets that require licenses. But then we must describe precisely under what conditions a system isn't restricted and under what conditions it is. The thought of a "computer use licensing structure," with different types of licenses for different sets, brings to mind a bureaucracy that will inhibit and complicate computer use. And given the widespread use of computers currently, whether such a structure could be put into place now is questionable. Furthermore, either all computer components would need to be controlled merchandise that required proof of licensing to purchase, plus severe penalties for bringing such merchandise into the country without a license, or any home hobbyist will easily be able to evade the controls.

One final difference is critical to understanding the aftermath of failures. In general, people involved in an automobile accident can be identified, either at the time—by looking at their licenses—or by their physical appearance. The same isn't true of security failures, or attacks, on the Internet. Indeed, anonymity is often considered a strength of the cyber infrastructure because it protects (for example) dissidents in repressive regimes. But that same strength becomes a weakness when we seek to identify other parties in electronic communications or incidents. The cyber infrastructure isn't yet as well defined, as well understood, or as mature as the automotive infrastructure.

Computers as Appliances

So, training people who use computers to understand security is highly unlikely to succeed. Alternatively, we can turn the analogy with automobiles around to ask: when the technology for computing has advanced to the point that using a computer *is* analogous to driving a car, what type of training will users and owners require? And what type of training will service center personnel require?

Answering these questions requires looking at how computing will be realized in such a world. People will need to be insulated from the complexities of computers, as they are from the complexities of automobiles. So, vendors will need to supply computers as "boxes" designed to perform the functions users want, with no additional software or modifications.

This suggests that vendors will need to supply several different types of systems, each with a preset configuration. One could be for people who do word processing, another for billing, and a third for both word processing and billing. People will buy these "boxes," plug them in, and use them. To enable extensions and expansions, each box will have defined a set of attributes describing what may (or may not) be added. Third-party software and hardware will similarly have a set of attributes with which they are compatible. By comparing the two sets, consumers can determine whether the third party merchandise is compatible with their system. No system can meet a person's needs precisely, but then no automobile can meet each family's needs precisely; "good enough" is the standard by which people will judge systems.

The trend of providing attribute information to provide a common language for describing systems exists in many fields. For example, a book is uniquely identified by its ISBN (International Standard Book Number). In the US, most products have a unique UPC (Universal Product Code). In the field of computer

security, most vendors of security products use CWE (Common Weakness Enumeration) or CVE (Common Vulnerabilities Enumeration) numbers to identify the security problems their scanners detect or their defenses protect against. The "attributes" I mentioned will probably evolve similarly to these mechanisms.

Where nothing "good enough" is available, people will turn to third parties to modify these commodity computers to meet their needs. These vendors will have the knowledge and skill to warranty their work and provide consumer with the desired system—without the user needing to know how it's done or how the system internals work.

The point is that the technology will be transformed into something that the average user need not understand. Consumers will simply have to know how to use those parts that affect them, such as a word processor. Furthermore, those components will be more reliable than most current programs and systems, simply because of the financial and temporal costs to vendors and consumers of failing to do so.

These changes will require two different types of training: that for users and owners, and that for vendors and service centers. The differences in training arise because the former focuses on use ("driving," in terms of our automobile analogy) and is most likely to be informal; the latter requires learning details of construction and implementation ("under the hood," to continue the analogy with automobiles), and will probably have more formal elements.

Using and Fixing the Computers

Consumers will need to learn how to use their devices. Interfaces will be designed to be intuitive, but whose intuition? Programmers' intuitions have produced interfaces that are notoriously difficult to use. More critically, peoples' intuitions differ. So, instruction manuals and tutorials will still be needed, and useful, to provide this instruction. They will be written for those who aren't computer-literate, much as manuals for running household appliances like televisions and telephones describe how to use the various functions. Consumers also need to be able to decide when something goes wrong and how to handle the problem. To a great extent, error indicators will provide this information, much as automobiles notify people of problems by maintenance lights, fuel gauges, and other indicators. But many problems that could occur are difficult to relay via a simple interface. For example, a fault on a television set could make people's skin appear green rather than brown, and no amount of adjustment will restore the original color. Then, people will need to call a repairperson or take the system to a service center for repairs. This means that consumers will need to be able to distinguish between "user error" and "system/program error." Currently, the distinction is ill-defined: is entering a string where an integer is expected a user error (bad input) or a program error (failure to check for bad input)? Part of the technology's evolution will be to clarify these ambiguities, and, in either case, the service center will be able to help.

The service center will take a holistic approach to fixing the computer. Thus, service center personnel will need training to analyze a system and a set of programs to determine whether the problem is with one particular component, a set of components, or their interaction. This will require them to discuss the problem with the customer, to determine exactly what the symptoms are and under what conditions it arises. Thus, some service center staff will need to be trained to extract the details from users who don't understand how the system works and could have erroneous preconceptions that lead them to not emphasize critical details and instead focus on less relevant ones. Once the problem is isolated, service center personnel must then determine how to fix it—or whether it can be fixed.

Should the (software) component be compromised (for example, a file containing malware had been read, resulting in the program file being infected), it might affect other components. Another problem might be internal to the component, such as setting incompatible configuration options. Rather than decontaminate the component or reconfigure it, perhaps the technology will enable the software to be discarded and reinstantiated. The cumbersome way to do this is to deinstall it; however, aside from the effort, this might not solve the problem unless all supporting components are also discarded and reinstantiated. An alternative would require vendor cooperation: simply embed the software in a virtual machine. Then, should the software malfunction or become corrupted, the VM could be closed and the software restarted. This eliminates any artifacts between uses.

The problem with this replacement approach is the loss of persistent data such as configuration data. People often spend considerable time configuring programs such as word processors or spreadsheets, and discarding the program and all associated components will also discard this data. Failing to discard this

associated data, though, could result in the new instantiation being corrupted. Thus, the workers in the service center must know how to handle situations like this—a topic that requires considerably more research.

Technology evolves; how people use the technology does as well. Most people don't care how the technology works—they care only that it does work. Current models are unsatisfactory because people must understand more of the underlying technology than is needed to perform their jobs. Either they must be trained to gain this understanding, or the technology—and its packaging—must evolve. The view of this evolution explored here proposes that the training and maintenance will rest in the hands of the vendors and service personnel and not the average user. Perhaps this is heretical, given the proposals for improved "security awareness" and similar educational programs. The thesis of this column is that any such efforts beyond the most basic—efforts that require users to understand how systems and defenses work as well as how to use them—will fail. People simply won't be interested, will lack the aptitude, or will lack the time to learn these matters. Any plan that fails to take this into account will itself fail.

References

1. K. Koscher et al., "Experimental Security Analysis of a Modern Automobile," *Proc. 2010 IEEE Symp. Security and Privacy*, IEEE CS Press, pp. 447–462.
2. T. Hoppe, S. Kiltz, and J. Dittmann, "Security Threats to Automotive CAN Networks—Practical Examples and Selected Short-Term Countermeasures," *Proc. 2008 Computer Safety, Reliability, and Security International Conf.*, Springer-Verlag Berlin pp. 235–248.

Matt Bishop is a professor of computer science at the University of California, Davis. His research interests include vulnerability analysis and denial-of-service problems, formal modeling (especially of access controls and the Take-Grant Protection Model), and intrusion detection and response. He has a Ph.D. in computer science from Purdue University. He is the author of *Computer Security: Art and Science* (Addison-Wesley, 2002). Contact him at bishop@cs.ucdavis.edu.

Abstract:

As technology advances, the ways people will interact with the technology changes to reflect the changes in technology. But most people do not care *how* technology works, but only that it does what it is supposed to do. Requiring them to learn how to secure their systems ensures that the systems will never be secured. Instead, the human element of securing systems must be transformed, along with the system.

Keywords: securing systems, transforming technology, human factors