TEACHING CONTEXT IN INFORMATION SECURITY

Matt Bishop

University of California at Davis

Abstract: This paper discusses teaching the application of technical idea through non-technical discussions, especially the use of puzzles to engage students. After discussing the need for teaching students to evaluate contexts in which decisions about computer security must be made, we suggest the use of questions and scenarios drawn from political science, history, and other humanities to force students to apply or derive principles of computer security to unusual and unexpected situations. Our experience shows that students find the process enjoyable, stimulating, and effective.

Key words: environment, judgment, instruction, computer security

1. PROBLEM STATEMENT

Software engineers used standard software engineering processes to develop software. They installed it in a state-of-the-art touch-screen system, and spent great care in making the interface easy to use by even the most naïve user. They used encryption to protect the results being transmitted to a central server, and powerful systems to provide the initialization and computation of results, and enabled an electronic audit log so results could be validated. Then they applied an ISO 9000 process to ensure good quality control, and had their product certified by both US Federal and state testing laboratories. Last-minute patches and modifications were made to ensure the systems worked. Yet, when deployed and used one day, the machines failed repeatedly. As a result, the company's right to sell its systems was withdrawn, and the company is being investigated for violating several laws. What went wrong?

The vendor sells electronic touch-screen voting systems (called DREs), and the server used in Clerk-Recorders' offices to total the votes. The problem that the vendor encountered was a failure to recognize the particular context in which DREs are used. The public, and the candidates, expect the systems to meet certain (nebulous) requirements, but three that people agree upon are: correctness of recording the votes and counting them (accuracy), anonymity of any individual's vote, and availability of the systems. Unlike standard systems, however, the DREs and the server are used only on Election Day. Failures of most types of software are annoying, but delays of an hour are not catastrophic. But in certain environments (notably medical and aviation), delays may be catastrophic, and vendors use high-assurance techniques to ensure that chances of failure are minimal—and even then, they usually provide backups to handle failure. If voting systems are not

available, or inaccurate, voters are disenfranchised, which—in a democratic or republican society—is unacceptable. The DRE vendor's error was in not realizing the social context in which its product would be used resembled the time-critical environment more than the ordinary-use environment.

The vendor's error is a common one. Consider the popular phrase, "securing cyberspace". The term "security" has too many meanings in this context. What are the requirements? They vary from industry to industry, indeed from organization to organization; the requirements for an academic institution, a company, a public charity, and a government agency all differ. The problem is that the techniques that exist, and are being developed, to secure systems are being deployed without adequate consideration of the context in which they will be used. "Are these methods appropriate and effective *for this particular environment*?" is a question heard all too infrequently.

A computer security course providing a basic education in various aspects of securing computers and information must also teach students when to use techniques, and—more importantly—how to analyze social, political, and cultural environments and contexts to determine whether a particular technique or technology is appropriate. The latter should not become all-consuming, to the detriment of teaching the technology and application of principles to computers, but it must be attempted in some measure.

Doing do has several benefits. First, teaching the non-technical aspects treats security as a holistic problem, rather than a purely technical one. This is reality. Mechanisms that are acceptable in some environments are not acceptable in others. For example, military environments can enforce procedural controls much more effectively than can academic environments. In the former, the chain of command provides a framework for stating and enforcing procedures. In the latter, questioning and autonomy are encouraged to a much greater degree. Procedures must be created and promulgated differently to be effective.

But not only mechanisms differ; policies do, too. Counting votes in an organization is different than counting votes in a public election, because the former can be re-run if there are irregularities. Worse, the policy may not be clear until *after* deployment, requiring both procedures and mechanisms to be adjusted accordingly. If 1% of the votes must be manually counted to verify any electronic vote counting system, does generating paper copies of the votes from the electronic equipment and counting those satisfy the statutory 1% requirement? This is a matter of law, not technology.

No purely technical treatment of computer security can produce competent security experts. Policy defines security, and the policy is given *ex cathedra*. Policies need to account for people. They must counter human foibles. They must describe security in a particular context or set of contexts that include environment, law, and expected user base. Hence, security is a human issue, and teaching it as a purely technological issue introduces gaps that technology cannot close.

The "best practices" rules gaining in popularity are good examples of this. Most describe how to secure a system. But many presume some definition of "secure" without stating it, and without explaining *why* the choices are made. If full log files cause overwriting, shutdown, or disable logging, the "best practice" rules should prescribe a choice appropriate to the goals of the system and its environment. If the log space is great enough so that the log cannot be filled between snapshots (and the log is cleared after each snapshot), then the setting is irrelevant. Otherwise, the policy must indicate which of the choices will give the greatest benefit (or do the least harm). People who have been exposed to the interaction of technical and non-technical matters will be able to make such judgments; those who have not will tend to follow the best practices without additional thought.

This judgment, this ability to apply principles and creativity in unusual or unexpected situations, is what computer security courses must encourage and teach. The key question is how to do so while keeping the students engaged, and supplementing the technical aspects of the instruction.

2. SOLUTION

To meet these needs, a computer security course can tie security principles into experience and practice by drawing examples from other disciplines. These examples should focus on five general areas.

First is teaching students to question assumptions. This technique helps one locate security flaws in a system, by uncovering gaps between the security mechanisms. However, it also uncovers assumptions about the environment. Returning to the electronic voting machine example, one assumption was that poll workers would hear a voting card being ejected, and thus be able to tell if a voter voted twice (because they would hear the second card being ejected too). But in practice, the noise at most precincts would mask the noise of the card being ejected. The assumption was not valid in the particular environment in which the machines would be used.

This suggests that all parts of a problem must be examined. This seems counter to the traditional "top-down" methods of analysis that are customary in computer science. In these methods, problems are decomposed into smaller problems, the smaller problems solved, and the solutions combined to solve the larger problem. If each smaller problem is solved *in the context of the larger problem*, then the traditional approach works. All too often, though, the smaller solutions ignore the context of the larger problem, and do not work properly. A classic example comes from warfare, in which one assumes that winning all the battles means winning the war. King Pyrrhus would disagree. After he beat the Persians in one of the innumerable battles between the Greeks and Persians, his army was so decimated that he said, "One more victory like this, and we are undone."

Examination of a problem as a whole sometimes leads to unexpected solutions. This offers the instructor the opportunity to emphasize the importance of looking at all aspects of the problem. A (possibly apocryphal) story of an attacker who repeatedly broke into computer systems makes this point. All sorts of technical measures were tried, but none succeeded. The defenders called the police, who tracked the perpetrator, a teen-ager, to another country that had no laws against this activity—so, naturally, when the police in that country were called, they declined to take action. The defenders were stymied, until a policeman had an unusual idea. He called the teen-ager's mother and told her what her son was doing. The attacks stopped immediately. The policeman's observation that this was a human problem, and his thinking that a primal human emotion (love of, or respect for, a parent) might solve the problem, was both astute and effective.

This emphasizes the need to consider human beings, both as individuals and as members of an organization. Security does not occur in a vacuum. Requiring users to authenticate themselves by providing urine specimens will not work in many societies because they violate customs of privacy. Similarly, expecting low-level employees to refuse instructions from a corporate vice-president who can fire them is quixotic at best. Security measures must take these inhibitions into account.

The most effective way to get students to understand these issues is to engage them by providing puzzles, and asking the students to brainstorm creative solutions. This has two benefits. First, it forces the students to think and speak up. This allows the instructor to guide the discussion to consider a variety of approaches without providing an answer. (As we shall see, some puzzles simply have no correct answer.) Second, the students often enjoy a short break from technical material, and a good puzzle will lead them to either discover some principle, or apply some principle, that also applies to technical material. In our experience, the students enjoy working with the puzzles, and sometimes suggest solutions or approaches—or complications—that the instructor had not considered. This also demonstrates the need for multiple viewpoints when considering security issues.

3. SELECTION OF PUZZLES

The two paramount rules for the selection of a puzzle are that the puzzle must engage the students, and the puzzle must illustrate some principle relevant to computer security. The instructor must know something about the students. If, for example, the students are from industry, one should concoct puzzles from business organizations and the world of commercial information technology. At a university, puzzles from the bureaucracy that students face are effective, as are puzzles drawn from other academic disciplines. For most students, current events can serve to supply puzzles, as can puzzles about break-ins and responses. People enjoy "war stories", and providing conflicts from a battlefield, a war, or politics, usually gets students very interested.

As to the second rule, the instructor can use the relationship between the virtual "cyber" world and the physical, "real" world. Principles of computer security derive from older principles. For example, the Principle of Least Privilege [Saltzer & Schroeder 1975] is a variant of the "need-to-know" principle so popular with governments and other organizations. The Principle of Separation of Privilege [Saltzer & Schroeder 1975] is a formalization of the idea of "defense in depth". Other parallels abound. The instructor can take advantage of this to illustrate the relationship between computer security and the human, organizational, environmental, and other constraints that students must consider.

Creating puzzles is straightforward, once one finds a topic. If the inspiration comes from a passage in a book or article, one can ask the students how the passage demonstrates a specific principle. A more open-ended method is to ask the students which principles the passage demonstrates, and why. A variant is to ask how to apply the contents of the passage to a computer system. The author prefers the open-ended approach for two reasons. First, if the discussion strays, the instructor can bring the discussion back to the points he or she wishes to make. Second, the students sometimes think of values or relationships that the instructor has not considered. The open-ended approach also encourages the students to speak more freely, as the problem is not so constrained as in the first approach. In the author's experience, encouraging creativity requires encouraging the students to express and consider ideas that sound crazy. Neils Bohr's comment that a colleague's idea was crazy, but not crazy enough to be true, is as reasonable in computer security as it is in quantum physics.

News stories and current events are also a fertile source for puzzles. Here the approach is slightly different. Rather than ask about general principles, the instructor can ask the students to put themselves in the position of the people involved in the incidents, or in the position of an analyst who is seeking to prevent or enable the actions in the incident, and say how they would act or what questions they would ask. The latter is particularly important. Students need to understand that security usually involves dealing with incomplete information, and part of what a good analyst does is asking questions. Taking this approach teaches the students how to analyze a problem, figure out what *additional* information would help them decide, and how to ask for it. When discussing fiction or historical incidents, this approach may fizzle. Either the students feel too remote from the events, or they can "look in the back of the book" to see the answers. But students are living in the times of current events and news stories, and easily relate to the problems.

4. EXAMPLES AND EXPERIENCES

What follows are some example puzzles in various areas of computer security. These were used in several undergraduate classes to spark thought and discussion among students, as well as to bring out points that the instructor wished to emphasize. We presented one puzzle at the

beginning of each class, and had the students discuss among themselves for 5 minutes or so; then, the class discussed their conclusions and ideas. The puzzles fell into four (broad) categories.

4.1 Questioning Assumptions

Saul Alinsky's books *Reveille for Radicals* and *Rules for Radicals* provided fertile ground for this category. Alinsky was a protégé of John Lewis, and delighted in organizing the poor and disenfranchised to force those in power to respond to their needs. His descriptions of tactics give insight into ways to attack systems, both political and computer.

An example passage is the following illustration of one of Alinsky's rules of tactics for an organizer:

> The third rule is: *Whenever possible go outside of the experience of the enemy*. Here you want to cause confusion, fear, and retreat.
>
> General William T. Sherman, whose name still causes a frenzied reaction throughout the South, provided a classic example of going outside the enemy's experience. Until Sherman, military tactics and strategies were based on standard patterns. All armies had fronts, rears, flanks, lines of communication, and lines of supply. Military campaigns were aimed at such standard objectives as rolling up the flanks of the enemy army or cutting the lines of supply or lines of communication, or moving around to attack from the rear. When Sherman cut loose on his famous March to the Sea, he had no front or rear lines of supplies or any other lines. He was on the loose and living on the land. The South, confronted with this new form of military invasion, reacted with confusion, panic, terror, and collapse. Sherman swept on to inevitable victory. It was the same tactic that, years later in the early days of World War II, the Nazi Panzer tank divisions emulated in their far-flung sweeps into enemy territory, as did our own General Patton with the American Third Armored Division.[3]

Those who attack computers act as Sherman did: they ask about the assumptions the defenders are making, and attack in ways that the defenders have not prepared for. Unless the defenders can handle situations they do not expect, they react the way the South did: confusion, panic, and collapse. Now consider security procedures for handling attacks. Applying the lesson from this passage to these procedures, students see that they must be prepared to depart from procedures as needed, and know how and when to do so. A second benefit is that the discussion can educate the students on why rigidity serves people ill, and flexibility well, in incident handling.

4.2 Holistic Thinking

Holistic thinking asks students to look at problems in context, rather than in narrow technical perspective. The following puzzle illustrates one approach to encouraging this mode of thinking:

> Microsoft spent February of last year teaching its programmers how to check their code for security vulnerabilities and how to introduce common security flaws. Yet many Microsoft programs still have security vulnerabilities. What problems do you think Microsoft encountered, and will encounter, in trying to find and clean up the vulnerabilities in its systems?

Initially, students brainstorm the technical problems that Microsoft faces. But those are relatively minor compared to the multiplicity of environments in which Microsoft systems are used. Vulnerabilities are defined in terms of the local policy, and Microsoft cannot build systems to satisfy all those policies. So Microsoft programs will continue to have vulnerabilities. Further,

[3] [1], pp. 127–128

Microsoft supports backwards compatibility on their systems. Fixing vulnerabilities may break this feature. What are the trade-offs?

The instructor can also point out the difference between security vulnerabilities and poor coding practices. Buffer overflows may indicate security vulnerabilities; they always indicate non-robust coding problems [Bishop & Frincke 2004].

4.3 Human and Organizational Problems

Sun Tzu's *The Art of War* and Niccolò Machiavelli's *The Prince* are excellent sources for problems of this type, although many news stories provide fodder as well. The point of these stories is that security must take into account people and organizations.

This passage from *The Prince* enables an instructor to illustrate how organizational problems affect security considerations:

> It can be put like this: the prince who is more afraid of his own people than of foreign interference should build fortresses; but the prince who fears foreign interference more than his own people should forget about them. The castle of Milan, built by Francesco Sforza, has caused and will cause more uprisings against the House of Sforza than any other source of disturbance. So the best fortress that exists is to avoid being hated by the people. If you have fortresses and yet the people hate you they will not save you; once the people have taken up arms they will not lack for outside help. In our own time, there is no instance of a fortress proving its worth to any ruler, except in the case of the countess of Forli, after her consort, count Girolamo, had been killed. In her case the fortress gave her a refuge against the assault of the populace, where she could wait for succor from Milan and then recover the state. Circumstances were such that the people could not obtain support from outside. But subsequently fortresses proved of little worth even to her, when Cesare Borgia attacked her and then her hostile subjects joined forces with the invader. So then as before it would have been safer for her to have avoided the enmity of the people than to have had fortresses. So all things considered, I commend those who erect fortresses and those who do not; and I censure anyone who, putting his trust in fortresses, does not mind if he is hated by the people.[4]

Technical courses rarely emphasize the importance of security officers obtaining and retaining the good will of the users and system administrators. Without that good will, the officers will spend more time dealing with recalcitrant and upset authorized users than they will with attacks from outsiders. With that good will, the officers will have many more people reporting suspicious problems, and system administrators who are receptive to adding, configuring, and applying security mechanisms. The instructor can use this to load into a discussion of organization techniques and processes to encourage this type of collaboration.

4.4 Thinking Out of the Box

Unusual problems demand creative solutions. The following story from *The Art of War* illustrates this point.

> If we do not wish to fight, we can prevent the enemy from engaging us even though the lines of encampment be merely traced out on the ground. All we need to do is to throw something odd and unaccountable in his way.
>
> Tu Mu relates a strategm of Chu-ko Liang, who in 149 B.C., when occupying Yang-p'ing and about to be attacked by Ssu-ma I, suddenly struck his colors, stopping the beating of the

[4] [Machiavelli], p. 69

> drums, and flung open the city gates, showing only a few men engaged in sweeping and sprinkling the ground. This unexpected proceeding had the intended effect; for Ssu-Ma I, suspecting an ambush, actually drew off his army and retreated.[5]

Asking the students to apply this to computer security draws interesting reactions. The key is to get students to think about how appearing to be weak, or unconcerned, can help improve security. After some discussion, this leads to the role of deception in computer security, a very fertile area—and one, in the author's experience, that the students enjoy.

5. CONCLUSION

Our experiences in using this technique have been uniformly good. In evaluations, students cite the "puzzle time" as a very enjoyable, educational aspect of the course. During the discussions, most of the students speak up—the most serious problem encountered is crowd control! This has the side benefit of encouraging students to ask questions during class, and makes clear to them that we encourage them to bring up ideas and misunderstandings even when the student thinks they are stupid or silly. Sometimes, those "stupid or silly" ideas suggest approaches and insights that would otherwise be overlooked. This applies both for the non-technical, judgment aspects of the course and for the technical aspects of the course.

Judgment is a critical facility for engineers and scientists. But the trend in teaching computer science, and other engineering and science disciplines, is towards mathematical, analytical rigor. Analytic understanding and rigor are essential to understanding these fields, but equally important is the ability to apply the techniques and technologies appropriately. These fields are sciences, but their application is also art. The art of computer security needs to be emphasized far more than it is in most courses. Using puzzles drawn from non-engineering disciplines helps this process immeasurably.

References

Saul Alinsky, *Rules for Radicals*, Random House, Inc., New York, NY (1972).

Matt Bishop and Deborah Frincke, "Teaching Robust Programming", *IEEE Security and Privacy* **2**(2) pp. 54–57 (Jan. 2004).

Niccolò Machiavelli, *The Prince*, Penguin Books, New York, NY (1995).

Jerome Saltzer and Michael Schroeder, "The Protection of Information in a Computer System", *Proceedings of the IEEE* **63**(9) pp. 1278–1308 (Sep. 1975).

Sun Tzu, *The Art of War*, Dell Publishing Co., New York, NY (1983).

[5] [Sun Tzu], pp. 27–28