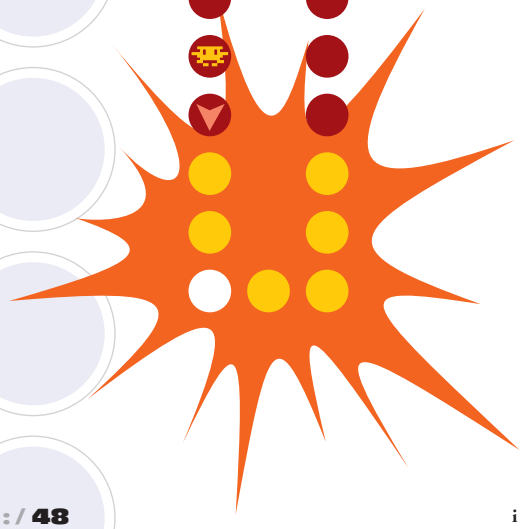
Firefighters and Engineers

Ka-Ping Yee | University of California, Berkeley | ping@zesty.ca

SPECIAL SECTION HCI & SECURITY

Computer security can be described in two different ways: keeping users away from dangerous things, or enabling users to do useful things safely. The former perspective is attack-oriented; the latter is task-oriented.

In the attack-oriented mindset, users trudge along a dark path through the jungle, fraught with perils at every turn: viruses, Trojan horses, spyware, and email scams are the deadly snakes and poisonous spiders of computer security. Like a guardian angel, from time to time security software swoops down to vanquish the enemy or stop the user from doing something stupid. This is the picture that the news media would like us to see, because it is dramatic and sensational. It is also a picture that suggests users should be fearful of using computers and need to buy security products for protection.



Users of today's popular computing systems do benefit from protection. But there is an important fact missing from the story: Essentially everything in software is invented. Whereas the dangers of a real jungle are unavoidable facts of nature, it is programmers who created the software jungle full of dangers and thereby fabricated the need for that guardian angel. Each danger is just as invented as the useful capabilities that software provides. To be sure, many of the invented dangers are now so entrenched in our computer systems that they will remain for a long time, but that doesn't mean they must always be there. We do not necessarily have to keep building systems that contain the same inherent dangers we have always faced.

In the task-oriented mindset, software is designed to carry out user tasks. Users communicate their intentions to their computers as they use them, often breaking down tasks into smaller steps, and the computer carries out each step in a way that most closely matches the user's specific tasks. Each step is implemented according to the principle of least privilege: Expose the user only to the minimal risk necessary to perform that step. In the ideal case, security becomes a consequence of not having programmed the computer with the capability to disobey the user, rather than of having identified and intercepted every possible attack.

The two mindsets correspond to two different roles that security professionals play. Sometimes they are firefighters, responding to vulnerabilities and incidents as they are discovered; sometimes they are engineers, designing and building tools and solutions for problems that users face. But let's not confuse the two by engineering software with a firefighter mindset. If we thought about furniture the same way we often think about software, desk lamps would burn rocket fuel and come with fire extinguishers strapped to them (not to mention a disclaimer of liability and instructions for using the fire extinguisher that no one would read).

What does this have to do with usability? Attack-oriented security tends to be difficult to reconcile with usability goals. In an attack-oriented design, the primary question is "Does it contain an attack?" That's a difficult question to answer correctly. When a prompt box requests confirmation to open downloaded files or activate macros, it delegates this question to the user-a question the user cannot possibly answer without substantial programming expertise and knowledge of the program code. When a virus scanner tries to make this determination, it consumes substantial amounts of time and resources, thus impacting usability. Yet it cannot definitively know which files contain attacks, since the definition of an attack depends on the user's intentions. Firewalls, Web site blockers, and other kinds of filters that lack information about the user's task are similarly guaranteed to provide incomplete protection.

Task-oriented security is where usability really has a central role to play. Enabling users to express their tasks more naturally and enabling computers to understand more accurately what users want are the bread and butter of usability engineering. And both of these are necessary to enable a computer to carry out tasks more safely, which directly yields better computer security. Designing software to do just what the user wanted—neither too little nor too much—also allows us to reduce or eliminate security warnings, security configuration settings, and their associated usability costs. Consequently, usability practitioners are likely to be more effective collaborating with security practitioners who are working in an engineering role rather than a firefighting role, though the former can be hard to find.

High-profile companies and projects tend to have organized firefighting efforts such as network monitoring, security response teams, software updates, and vulnerability reports. Much of what "security engineers" do is actually firefighting. More attention needs to be devoted to true engineering efforts. This is not to say that firefighting is unimportant. We need firefighters; they are our last line of defense. But it's time to draw a line between old software and new software; between the reactive and the proactive; between firefighting and engineering. Constantly rushing to fight yesterday's battles won't yield long-term improvements. We need to start thinking ahead and shifting the focus toward a task-oriented perspective if we want software to be better, safer, and more usable in the long run.

ABOUT THE AUTHOR

Ka-Ping Yee is a PhD student at the University of California, Berkeley who is developing a verifiable user interface architecture for voting machines. His other interests include interaction design for secure systems, capability-based security, information visualization, and computer support for activism, argumentation, and decision-making. Ping is from Winnipeg, Manitoba, Canada.