2010 National Computer System Security Award
Presented at the IEEE/SP 2010 Symposium on Security and Privacy, Berkeley, California
Monday, May 17, 2010
Acceptance speech by Jerome H. Saltzer, MIT

The following text was prepared in advance for the acceptance speech. The talk as actually delivered was not read verbatim, but it did follow this text fairly closely.

Thank you for that kind introduction, Tim. I must confess to being a little puzzled about why I am here, since I haven't really done very much in the area of security and privacy in a long time. I am responsible for only a small number of very old published papers on the topic, usually as a co-author, and some of them not even refereed. Since much of this audience was born after those papers were published, let me start by relating a bit of history.

Back in 1963, I was a grad student working for Fernando Corbató, who was the driving force behind the Compatible Time Sharing System and Multics. Corby pointed out that when a computer is shared it is important to keep the users out of each other's business, and the design for CTSS included some features with that purpose. So Max Smith and I began poking around to see just how secure it was. It didn't take us very long to find quite a few security holes. Some were the result of simple programming errors, but more were because system programmers hadn't quite absorbed the goal into their thinking.

A year or so after that, work began on Multics. Since this was to be a completely new design, we set out to apply a new doctrine—that we should design security into the system from the beginning. The doctrine had considerable encouragement from the senior designers, not just Corby, but especially Ted Glaser, who had come to us from Burroughs and was consulting for the National Security Agency at the time, and Vic Vyssotsky of Bell Labs, who seemed to have a built-in intuition about almost anything to do with computer systems.

So we indoctrinated all the designers and programmers, and we included lots of security mechanisms—virtual memory, memory segments with permission bits, rings of protection, access control lists on files, access control lists on access control lists, and so on.

What we learned was two things. First, users couldn't figure out how to use most of the security mechanisms. So what they did was configure them to minimize interference with useful work. Second, the system was ***not*** secure--it still had implementation bugs, design mistakes, and system programmers still didn't realize many of the implications of their designs.

Since I was teaching a course on computer systems, and we needed class notes for the discussion of security, Mike Schroeder and I began writing down everything we could think of about the subject, and in 1975 we published those course notes as a tutorial paper in the Proceedings of the IEEE. We organized our thoughts around a handful of design principles, most of which were either ancient wisdom or suggested by other people—Auguste Kerchkoffs, Bernie Peters, Ted Glaser, Roger Needham, Roger Schell, Gerry Popek. All we did was pull them together in one place and give them catchy labels.

That was 35 years ago. Despite much intervening research and development, few deployed systems seem to offer much by way of security. Instead we are faced with bot armies, denial-of-service attacks, flash worms, phishing, buffer overflows, and untraceable spam. And lots of security researchers scratching their heads trying to figure out what to do about it.

So much for history and where we are today. How about the future?

Are things getting better? Can we finally relax and expect to see existing security bugs gradually getting fixed, and fewer new ones arriving? Perhaps subcontract security to a handful of paranoid specialists so that we can move on to other things?

Unfortunately, I'm afraid the answer is NO.

The problem is the rapid rate of technology change. Technology continues to improve so rapidly that legions of elves rushing to exploit it create new vulnerabilities faster than the rest of us can discover them and root them out. We can see this lesson in the experience of the personal computer and the Internet. Rapid technology improvement is what allowed the PC to become a household item before it was secured, and it also allowed the Internet to take off so rapidly, freezing its protocols in an insecure state.

In the 1990's the problem was single-user PC's being attached to a network. Today it is web interfaces to SQL databases and code injection. It is active data such as images and PDFs that harbor malicious code. It is running JavaScript programs in your browser, inside a sandbox of dubious trustworthiness and written by a contract application designer who started work just last week. It is FaceBook slipping up yet again. Tomorrow it will be something else, invented by yet another of the many well-meaning elves who create and distribute code in vast quantities.

The bottom line seems to be that you have assurance of continued employment as a security wizard.

Peter is starting to glare at me—it is time to wrap up.

One thing I have found gratifying over the last couple of decades is the number of bright young computer science students who are taking a strong interest in computer security and privacy. Many of these now have their Ph.Ds and are undertaking their own research and accomplishing interesting things. I hope that this award can act as an inspiration to that younger generation.

I want to express my appreciation to the NIST and NSA committees that selected me for the award, and also to the audience which is patiently sitting through a long round of acceptance speeches. Thank you, and good evening.