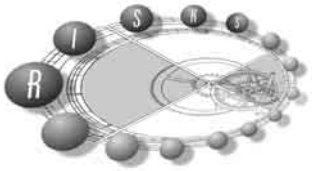

Security by Obscurity

The belief that code secrecy can make a system more secure is commonly known as security by obscurity. Certainly, vendors have the right to use trade secret protection for their products in order to extend ownership beyond the terms afforded under copyright and patent law. But some software systems must satisfy critical requirements under intensive challenges, and thus must be trustworthy. The following scenarios illustrate the limitations of the myth of security by obscurity.

The Ostrich. Metaphorically, many people think (falsely) that ostriches put their heads in the sand in the belief that they are invisible. Some designers think that by restricting access to their system code, exploitable vulnerabilities will not be exposed. The fallacy in this line of reasoning was evident in Matt Blaze's 1994 discovery of a flaw in the Escrowed Encryption Standard (Clipper) that could be used to circumvent law-enforcement monitoring (www.risks.org, risks-16.11 and 16.12). Surprisingly, Blaze's method allowed for even easier access to secure communication through the proliferation of Clipper products than was previously possible, without access to any keys, backdoors, or weaknesses in the encryption algorithm. (Hiding cryptographic keys is of course necessarily a form of security by obscurity.)

The Emperor Has No Clothes. A fabled trusted entourage agrees with a foolish assertion because each observer fears retribution. Software is not like Coca-Cola®, where for decades a handful of key employees have been trusted with a secret recipe and production process. Many computer systems are constructed in environments where a host of developers, debuggers, integrators, evaluators, and end users (with shared or possibly conflicting interests) require access to the proprietary product. Each of these individuals or agencies (collectively and individually) may hold the "keys to the kingdom," not only because they have knowledge of some or all of the secret code, but because they may be aware of limitations and constrained from revealing or sharing this information. Also, organizational culture may discourage whistleblowing, even when dire consequences are possible. This happened in both space shuttle accidents, where the O-ring and debris damage problems were known within the NASA community before the fateful missions.

PAUL WATSON

I've Got a Secret. The ease with which digital files can be transmitted (willingly or inadvertently) compounds the software secrecy problem. Earlier this year, several proprietary voting system program files were discovered on a subcontractor's unsecured FTP site. The vendor (Diebold) argued the software subsequently reviewed at Johns Hopkins represented "a very small percentage of the entire code needed to conduct an election" (www.diebold.com/election.htm). Of course, this does not excuse the lax security that allowed the code to be downloaded from the Internet in the first place.

The Shell Game. Here, a trickster uses slight-of-hand to keep an object from view. In the preceding voting system example, the Johns Hopkins analysis team found numerous security flaws in the code, one of which involved the use of a vulnerable DES encryption protocol, along with a hardcoded key in the source file (www.avirubin.com/vote.pdf). Diebold defended its system in a rebuttal report, claiming the examined software was "an older version" that had "passed rigorous functional tests and reviews" (www.diebold.com/checksandbalances.pdf). However, many election equipment tests are performed in secret, thus making it impossible to ascertain the level of rigor applied. One such reviewer, Douglas Jones, had served on Iowa's Board of Examiners and, based on his analysis of a federal test report, had asserted to Global (Diebold's predecessor) in 1997 and the House Science Committee in 2001 that inappropriate key management was being used with the firm's products (www.cs.uiowa.edu/~jones/voting/dieboldacm.html). It will be difficult to know whether such problems have been adequately resolved, because Diebold plans to continue its closed-source practices.

As noted here last month, open source by itself does not provide a solution to these problems. Risk assessment, examination, and testing appropriate to deployment settings are fundamental to security assurance—which should not be hampered by vendors' refusals to disclose critical components where a need to know can be demonstrated. Furthermore, customers should not cling to the false hopes of security by obscurity.

Rebecca T. Mercuri (mercuri@acm.org) is a research fellow at Harvard University's Kennedy School of Government and author of *Communications'* Security Watch column. **Peter G. Neumann** moderates the ACM Risks Forum (www.risks.org).