Probing End-User IT Security Practices—Through Homework

Student exploration of individual security practices yields instructive insights for campus IT environments

By **Sean W. Smith**

At Dartmouth College, I teach a course called "Security and Privacy." Its early position in the overall computer science curriculum means the course needs to be introductory, and I can't assume the students possess an extensive computer science background. These constraints leave me with a challenge: to construct meaningful homework assignments that expose relatively inexperienced students to the security and privacy issues pervading our electronic society.

Rather than tackling the highly technical issues buried deep within the infrastructure, I started with assignments that focused on the edges: the IT that students use and encounter in their daily work. Besides making the assignments accessible, this focus made the security issues tangible—the risks affect what the students use now and what they will use when they go out into the real world.

In the past few years, we've looked at issues involved in hidden data in Word documents, password practices, and the paradigm shift from tethered to wireless networking. What students discover, besides proving instructive to them, can teach the rest of us in campus IT environments about security issues.

Campus IT Background

Dartmouth currently has 5,500 students and a history of aggressively incorporating information technology into student life. University-wide e-mail, known as "Blitz," has been in place since 1985; a stroll through campus common areas and coffee shops reveals rows of public-access workstations, with students queued up to check their e-mail. Students use a variety of e-mail clients, including several Web-based ones. High-speed Ethernet—mostly switched—reaches dorm rooms, classrooms, and every library desk. In the past few years, a wireless LAN has also blanketed the campus, so strolls also reveal students with laptops checking e-mail—sometimes while they are in class.

A campus-wide name-and-password system handles authentication for e-mail and other services such as registering for classes, checking grades, and (for faculty and staff) accessing student records; even our pilot public key infrastructure on campus uses this system for initial registration. To avoid sending plaintext passwords, the authentication system adopted a random number exchange technique from AppleShare: the client concatenates the ASCII encoding of the characters in the user's password to form a digital encryption standard (DES) key, and then uses this key to encrypt an 8-byte random challenge from the server. (If the password is shorter than eight characters, the client pads it with 0-filled bytes.) Some campus clients and applications support Kerberos instead; others only support plaintext passwords.

Find the Hidden Data

Microsoft Word has become our society's universal communication medium. People turn their ideas into written communication through the Word program and expect recipients to view their content through Word. In users minds, the pile-of-bytes that constitutes the Word file *is* the virtual piece of paper they see when they open it with Word.

Except, of course, it is not—the Word file contains all sorts of additional data, some visible through the Word program, and some visible only through a binary editor. Users think they're sending the virtual piece of paper, but in fact they're sending much more. (Byers[1] recently published a limited examination of some of these issues; Zalewski[2] gives a much more amusing catalog.) To drive this point home, I regularly give the homework assignment to find in-the-wild Word documents that give interesting examples of this mismatch between the user's mental model and reality. I show the students how to use emacs (a standard text editor in the programming world) in hexl-mode (which lets the user examine files that consist of arbitrary binary data, not just ASCII characters. I demonstrate that many of the interesting nuggets are apparently UTF-16 (ASCII characters interspersed with 0-filled bytes)—so string searches that don't take into account these 0-filled bytes won't find the strings—and then send the students off to explore Word files.

Empirically, we've seen hidden data show up in four forms: apparently random data grabbed from other applications; internal Word structure; deleted data that is preserved via "fast save" (which older installations enabled by default); and revision history (viewable through Word's Track Changes feature). Students have found good examples of all four.

- *Random data.* A take-home examination (from another department) had pornographic URLs embedded in the binary data that constituted the Word file—but of course, these URLs don't show up when one views the file through Word. Similarly, a memo from a dean on plagiarism contained some unrelated e-mail to that dean.
- *Internal structure.* A student discovered his Word installation installing a virus in each document. Embedded usernames and pathnames revealed interesting histories behind administrative memos, student government documents, and resumes.
- *Fast save.* The cover letter a senior e-mailed to a prospective employer contained the names and addresses of the prospective employers the senior had previously contacted.
- *Tracked changes.* A research paper from the IBM T. J. Watson Research Center contained a completely different paper (by different authors); apparently, the author formatted his paper by downloading another and cutting and pasting his material over the original.

A student who went off into the real world recalled this homework exercise when a client company sent a PowerPoint presentation with some bottom-line figures cut-and-pasted from an Excel spreadsheet. It turned out the binary file that constituted this presentation contained the entire spreadsheet, with much presumably confidential information.

As another privacy-related exercise, my students also constructed Web pages that carried out Felten and Schneider's cache-timing attack[3] on Web browsers. By including elements from a selected site and timing how long it takes the victim's browser to render them, a malicious Web page can determine whether these elements are in the victim's cache and thus whether the victim has visited that site recently.

Social Engineering

As with many other enterprises, much of Dartmouth's IT security depends on passwords. Early 2004 brought reports that U.K. businesspeople will reveal their passwords in exchange for chocolate.[4,5] One report suggested that the chocolate reward wasn't even necessary.[6] Such an explicit revelation about the weaknesses of the foundation raises questions about the overall effectiveness of security technology.

Curious about these findings, the students asked if they could try some social engineering attacks on password security for their next homework assignment. Discussion with the campus CIO's office quickly set some guidelines, and they were off.

One team—we'll call them Alice and Bob—decided to extend the U.K. approach by offering a small plastic dinosaur or squirt gun, as well as chocolate, in exchange for a user's password. Those participants who would also reveal a friend's password earned two prizes. Eighty percent of those asked admitted to giving their real password. (It is interesting to note that, despite the mixed genders of the participants, Alice had a 100 percent success rate.) The team did not observe a correlation between success rate and graduating class; however, participants who rated their own computer skills highly were less likely to reveal their passwords than participants who felt their skills weren't as sharp. Fifty-seven percent of the participants admitted to having previously shared their passwords with someone else; of these participants, 95 percent gave their passwords to Alice and Bob. Alice and Bob observed that the prizes did not seem to sway participants, but were fun anyway. A plethora of plastic dinosaurs remained in the computer labs for finals period.

One student (call him Carl) tried a solo project. Rather than asking users in person, he send out e-mail at random, asking participants to visit a Web page that offered his questionnaire. This page asked participants first to authenticate and offered a link that took them to a spoof of one of the common Web-based mail sites on campus. Using a known flaw in Microsoft's Internet Explorer (IE), Carl constructed the spoof to appear to show the correct URL for participants using an older version of IE. Carl's spoof didn't actually collect the password, but simply registered if the user started typing into the password field. Eighty-three percent of Carl's participants tried to give their password to his spoofed site, although only 36 percent had the browser vulnerability. Of those who did not have this vulnerability, only 3 percent noticed the wrong URL.

Another student ("Doug") also tried a solo project. Like Carl, he offered a Web survey; however, Doug offered his Web survey in person (through a wireless laptop) and wrote a back-end script to verify the entered password-user ID pair with the campus authentication system. Doug hosted his survey page on a personal site within the dartmouth.edu host namespace. He used nice fonts and logos to make it look official and Secure Sockets Layer (SSL), through a meaningless self-signed certificate, to make it look secure. Ninety-three percent of the participants—including two faculty members—entered valid passwords. Some duped participants claimed that they thought the site was secure, since it had an https URL and a dartmouth.edu hostname. Others reported that they thought nothing was amiss, since campus health services routinely sends out electronic surveys that require authentication. One suspicious user distrusted Doug's laptop and so carried out the survey from his own machine—and either didn't notice or didn't worry about the browser's warning regarding the self-signed certificate. Another suspicious user gave someone else's name and password, which Doug's script validated as indeed correct (so don't lend your password to friends clever enough to be suspicious about roving security students). The faculty participants demanded to see Doug's student ID before taking the survey. They still gave Doug's site their real passwords, and afterwards, they could not remember Doug's name.

Rather than trying social engineering, "Eric" took a technical approach. Cryptographically inclined readers might notice technical weaknesses in how the campus authentication system uses passwords as DES keys. Brute-force searches of the 56-bit DES keyspace are feasible in reasonable time on highly specialized equipment. As a result of using a concatenation of ASCII encodings (where the most significant bit is 0) as a DES key (where the least significant bit in each byte is considered parity), however, our system, like Apple's before it, has an effective keyspace of only 48 bits. By eliminating unusable characters, Eric pared this space down even further. He then designed a bitslicing implementation of DES that exploits the AltiVec SIMD capabilities in the G4 and G5 CPUs in newer Macs and estimated that four dual 2 GHz systems could search the keyspace in three weeks. By itself, this does not enable a practical attack—Eric would have to spend three weeks on a challenge-response for each password he wished to recover.

Should a user choose a short password, brute-force attacks become trivial. (Choosing a password from the dictionary also makes hackers' attacks easier.) Unfortunately for my students, the campus system no longer permits such bad choices.

Wireless Attacks

As noted, our campus has jumped on the wireless bandwagon. Our wireless network is open, secured by a "secret" SSID and providing no encryption. We have seen, however, that users do not fully appreciate the paradigm change that wireless brings. Computing practices that were reasonable in a tethered network—particularly in a switched network, where a student cannot just put her network card in "promiscuous" mode and listen—become questionable when the wires go away.

One example is e-mail. Most of the clients for our campus's homegrown e-mail system do not support encrypted connections. On a switched network, the risk is minimal. But in an open wireless environment, users are sharing their e-mail with everyone within radio range. To demonstrate this risk, I have my teaching assistant read e-mail during class—except the e-mail is not his. He uses Kismet to capture packets and Ethereal to reassemble the streams; he also quips that WEP (Wired Equivalent Privacy, a security protocol) is merely a speed bump when using the appropriate tools. (Because reading others' e-mail violates campus policy, I do not assign this exercise to the students.) We always see interesting content.

I've suggested that, like Google and its lobby display of the most recent searches, our university set up some lobby displays of the most recent plaintext e-mail plucked out of the ether. The administration hasn't yet approved this idea.

While exploring password hacking, another student ("Fred") noticed that, on the wireless network, he could observe the campus authentication server sending a random challenge to a nearby user's client, and then observe the client responding. Fred further noticed that the duration of the DES operation was proportional to the password length (perhaps because of the 0-filled bytes—this bears further investigation). Thus, by timing the interval, Fred could determine which users had passwords short enough for a quick brute-force search; by recording the challenge-response pair for such users, Fred could carry out that search. Going to wireless gave him easy access to the necessary data. However, since the only users on campus with short passwords were high-level faculty and administrators who chose their passwords before the length limit was in place (and who have resisted pleas to choose new passwords), we decided against pursuing this experiment further. Carrying it out would also have required getting an effective performance estimate for the victim's machine, which might not be trivial.

In another cryptographic side-channel exercise, we implemented Paul Kocher's timing attack[7] on RSA (the de facto standard public-key cryptosystem) to demonstrate that simply using a private key can leak information about the key.

"George" used the open nature of the wireless network to transform long keyspace searches from a curiosity to something more devastating. George observed that the mail clients did not authenticate the campus authentication server. He also observed that his wireless laptop could tell when another user's wireless client requested a random challenge from the server—and that his laptop could easily forge a reply that, with about 50 percent probability, would reach the victim's machine before the genuine reply would. George precomputed a dictionary of how possible passwords would encrypt a random challenge of his own choosing—and then used the wireless network to provide this challenge to the victim, whose response he used to look up the password. In theory, this trick could also be carried out with a more difficult domain name server (DNS) attack, but wireless makes things easy. One wonders what other types of spoofing the wireless network makes easy.

In an open wireless environment, users are sharing their e-mail with everyone within radio range.

Parting Thoughts

Although I've told these tales with a lighthearted tone, I have a more serious message to impart. Universities and other enterprises spend a great deal of effort securing their information infrastructure. Whether this effort provides effective security depends on whether it meshes with how end users think about and use the systems. Thanks to the circumstances of trying to construct meaningful security lessons for an introductory course, I gave my students assignments that ended up probing the user end. They uncovered

- differences between users' mental models of a Microsoft Word document and its reality;
- dangerous user practices regarding passwords; and
- how safe user practices can become dangerous when a network switches to wireless.

The interaction between humans and security technology is a field only beginning to emerge.[8] This area deserves our attention—in research, in education, and in practice. *e*

Endnotes

1. S. Byers, "Information Leakage Caused by Hidden Data in Published Documents," *IEEE Security and Privacy*, Vol. 2, No. 2, March/April 2004, pp. 23–27.
2. M. Zalewski, "Strike that out, Sam" Web site, March 2004, <http://lcamtuf.coredump.cx/strikeout/> (accessed September 3, 2004).
3. E. Felten and M. Schneider, "Timing Attacks on Web Privacy," presentation at the ACM Conference on Computer and Communications Security, 2000, <http: //portal.acm.org/citation.cfm?id=352606&coll=GUIDE&dl=ACM&CFID=26585685&CFTOKEN=67764559> (accessed September 3, 2004).
4. "Passwords Revealed by Sweet Deal," *BBC News*, U.K. Edition, April 20, 2004, <http://news.bbc.co.uk/1/hi/technology/3639679.stm> (accessed September 3, 2004).
5. M. Wagner, "Will Trade Passwords for Chocolate," *Security Pipeline*, April 2004, <http://www.securitypipeline.com/news/18902074> (accessed September 3, 2004).
6. Steve Barnett, "Biometrics: Get Ready to Destroy All Passwords," *Bios: Technology for Business*, January 20, 2004, <http://www.biosmagazine.co.uk/op.php?id=81> (accessed September 3, 2004).
7. P. Kocher, "Timing Attacks on Implementations of D-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology—Crypto '96: 16th Annual International Cryptology Conference, Santa Barbara, California, August 18–22, 1996 Proceedings*, N. Koblitz, ed. (New York: Springer Verlag, 1996).
8. The Yahoo security group, HCISEC: Computer Security and Usability, <http://groups.yahoo.com/group/hcisec/> (accessed September 3, 2004).

Sean W. Smith (sws@cs.dartmouth.edu) is Assistant Professor, Department of Computer Science, Dartmouth College and Director of the Cyber Security and Trust Research Center at the Institute for Security Technology Studies.