Password Policy Purgatory

Stephen Farrell • *Trinity College Dublin*

At present I, and I'm sure you, have a lot of passwords. Too many, in fact, and it's getting worse rather than better. Many authentication schemes are, of course, more attack-resistant than simple passwords, such as token-based schemes or those based on public-key infrastructure (PKI). However, from users' viewpoints, if they have to enter a password, passphrase, or PIN, several issues arise with managing those values. In this article, I consider the use of passwords and equivalents from the end-user perspective, looking at how their usage interacts with administrator-enforced password policies. As we'll see, there's room for improvement, and policies could take a more realistic view of the current situation users face.

Password Categories

Many passwords have an associated user or account identifier, but for the sake of brevity, I ignore those in this column, even though they also require management and have potentially significant privacy implications. I'll start by categorizing the passwords I personally use, with categories reflecting the type of system to which the password is sent, rather than that system's relative importance, the authentication mechanism used, or the domain to which the password is presented (the categories aren't significant, other than for organizing this column).

Aside from the computers I carry around that require authentication (usually one of three or so, depending on what I'm doing and where), I log in to perhaps a dozen hosts, where even though I can, and do, set up private-key-based authentication (Secure Socket Shell [SSH; www.openssh.org]), passwords still lurk in the background that I occasionally need to use — for example, when "hopping" from one host to another where I've not set up the source host's SSH public key at the destination. Let's call these *login passwords*.

I also have a couple of user-specific network access passwords required for wireless or wired access, some of which overlap with the set of login passwords, and some of which are specific to this purpose. In my case, those that overlap are generally authenticated, ultimately by a Microsoft ActiveDirectory[1] domain. Let's call these *network access passwords*.

In addition, I have several passwords for home use, including the DSL router/wireless access point (and its security key), as well as print and file server administrator passwords. At work, I also sometimes deal with sensor networks and so must store passwords for deployed devices. Let's call these *device passwords*.

Then, there are application-protocol-specific passwords, from outbound HTTP proxies to mail servers (for both IMAP[2] and SMTP[3]) and various voice-over-IP clients. In these cases, I generally have an application client that remembers the password and presents it using the relevant protocol. Let's call these *protocol passwords*.

Some passwords are service specific, mainly those required for Web site accesses, but I'm sure I've set up a PGP private-key password for secure mail as well, although given how useful signed mail is — and encrypted mail being such a rarity — I've long forgotten that one. Some of these passwords protect highly important services (such as banking), some absolutely trivial ones (such as room booking), and some are in between (for example, proposal submissions), thus increasing users' difficulty in sensibly managing the set as a whole. But in any case, let's call these *service passwords*.

Finally, I need a fistful of PINs for credit and debit cards and a bunch of door codes to get to places where I can then type passwords

 1089-7801/08/$25.00 © 2008 IEEE

before doing something useful. I also, of course, have a PIN for my phone. Generally, these don't have an associated user name, and PIN or code entry is sufficient authentication. Let's just call these *PINs*.

In addition to all of these, I also manage several mainly protocol and service passwords for family members and a few login passwords for colleagues on systems that I administer. And as is probably typical, I use only the default/minimal password policies that apply for each one.

Table 1. A password snapshot.

Category	Count	Known	Examples
Logins	10	6	Laptops, host systems (including root accounts)
Devices	5	0	DSL router, home print/file servers, sensor nodes
Network access	4	0	LAN, WLAN, ISP, and so on
Protocol	14	1	Outbound HTTP proxy, IMAP, Jabber, Skype
Service	21	4	Web sites with passwords stored outside the browser
PINs	7	4	Bank cards, door access codes
Total	61	15	

Number of Passwords

Table 1 shows how many of each type of password I currently maintain. The "count" column lists the number of potentially different passwords, but not all have unique values (although because I'm a security person, many are, in fact, unique). Where a single password can fit into more than one category (such as IMAP and webmail), I've only counted it once.

The "login" category counts only systems for which I have different accounts, so hosts with centrally managed accounts count just once.

The "network access" category doesn't include one-time use wireless/hotel network passwords, but only those that I regularly require, or that a device I manage regularly requires. In the latter case, even though typing the value in question is very rare, I do still need to store the value to be able to recover from device failure or add new devices to a network.

The "service" category doesn't include passwords that I've only ever stored in a Web browser, but it does include passwords required for Internet banking and for Web sites that I might need for work purposes even if I change my laptop. It doesn't account for all the nonsense Web accounts required for hotel bookings, car rentals, and so on. My Web browser currently stores almost 200 such password entries, many of which overlap — for example, due to different URLs on the same host requiring sometimes the same (and sometimes different) account information. I also didn't include mail list archive passwords here, other than for lists that I administer.

The "known" column indicates those that are unique and that I can remember without having to look them up on a list. When I made up the table, I was actually surprised by how many I could remember. The literature[4] would indicate that I probably remember more than is common, again perhaps because I've worked in IT security for so long.

In my case, I store most of these passwords in plaintext files or on my phone. I don't encrypt those files because doing so would decrease portability (I've switched operating systems several times) and also because that would add only a small additional barrier. I assume that if I were the target of a sophisticated attacker, then a brute-force attack would in any case succeed, and a vandal or laptop thief probably wouldn't care. However, I'm as careful as I can be to not send passwords in clear over a network connection, except where that's unavoidable, because I consider network-based identity theft attacks a sufficiently serious concern (even though identity theft is more likely to involve a compromised server or laptop).

Seven of the passwords in the table aren't stored anywhere except in my head; those are mainly related to banking but also include my most frequently used login passwords.

And in case you're in any doubt, these numbers are the main justification for this article's title.

Password Policy Issues Arising

I use many of the passwords I counted in specific domains, each of which defines its own password policy. Let's look at just a few of the issues arising from the increasing size of such password collections.

Note, first, that everyone involved here experiences difficulties — domain administrators have a hard time letting users be careless with passwords because they'll get part of the blame for any password-based breaches. So, they understandably want to define and enforce rigorous policies but often ignore the fact that all "their" users might take part in many password domains and manage many other individual passwords.

From the user's viewpoint, we must often adhere to different and possibly mutually incompatible password policies, even though the human memory is a finite resource. One example is that some password policies restrict the ability to reuse old passwords, presumably on the basis that it limits exposure if

a given password is compromised. However, users will often[4] reuse the same small set of passwords in different domains, so in reality, when a current or old password is compromised in one domain, it likely puts other domains at risk. This reduces the utility of disallowing reuse.

Forced Change

Wayne Summer and Edward Bosworth have recommended password policies and provide some nice examples of the legacy from which we must evolve.[5] Although it might appear that I'm criticizing their recommendations, they were probably state of the art at the time, when many users had to remember only a few passwords for work and before the current (fashionable, but entirely justified) emphasis on usable security.

In any case, one recommendation — that organizations still, unfortunately, include in their policies — is a "maximum password age of 45 to 60 days," justified on the basis that a cracker will have had time to compromise any human-memorable password in that time. However, if even relatively few passwords are managed in this fashion, users will most likely end up choosing weaker or tightly related values, which nullifies the effect of forcing the password change, and, we might imagine, increases support costs. With $O(60)$ passwords, if 10 percent must be changed every two months, that's roughly one password change per week, which is clearly undesirable from the user's viewpoint.

The lesson here is mainly that administrators need to be aware that users have too many passwords to remember; they should take into account the user's overall burden and not behave as if their systems were the only ones with which the user interacts.

A second problem with such a forced-change policy is that it also ignores technological development: if password policies are revised only every two years, for example, then Moore's law or the application of distributed computing are both working to the putative attacker's benefit, whereas no likelihood currently exists of a Moore's law type increase in the capacity of human memory.

The argument here (I wouldn't call it a lesson) is that password policies should be driven by what the users can accept, not by an attacker's capability (all passwords being cryptographically weak, in any case). Administrators should also take into account the likely duration for which a given policy remains in force.

Password Reuse for Different Applications

Some systems also reuse passwords in ways that can conflict with a user's expectations. For example, one ISP I've used offered a voice-over-IP service and let you create a new identity for that purpose, but forced the initial password for the new identity to be the same as your primary ISP account password, which is generally only for account management. They also sent an email notification containing that password. This type of reuse (without offering the user a choice) can result in a password value that the user regards as sensitive because it's exposed in an unacceptable manner.

The lesson here is that administrators should consider that users might manage sets of passwords in ways that the administrator hasn't considered and offer a range of choices with very carefully considered defaults.

Too Much Authentication

Yet another policy pitfall that I've occasionally seen is the overuse of authentication, in particular for outbound HTTP proxy authentication. I know of one domain that provided the same initial password to users for proxy authentication, logins, and mail access via IMAP. Although it did offer ways to change each of these, it was never clear (to users) when changing one password value implied a change in others. In fact, even when different applications were coupled so that password changes were synchronized, significant latency sometimes resulted because of database synchronization issues or because the different applications checked the password at different times. In addition, although almost all IMAP use was secured with SSL, proxy authentication forced users to send their passwords in the clear over the network (although almost always on a switched LAN). In this case, I would argue that a better policy would have been to remove the requirement for outbound proxy authentication, which would result in both less confusion and increased security by avoiding transmitting cleartext passwords.

What we can learn from this is that administrators should be careful of password handling schemes that are similar to, but aren't in fact, real single-sign-on (SSO) systems. In a real SSO environment, all user authentication should use the "native" SSO mechanism for authentication (for example, SAML or OpenID),[6] and, for legacy applications, the SSO system should manage cleartext passwords as encrypted attributes so that most users wouldn't need to see or remember them.

Password Sanity and Reality

One Web site that appears to be on the user's side in this argument is the nicely named "Center for Password Sanity," (www.cryptosmith.com/sanity/), which goes into more detail and echoes several of my arguments. Shirley Gaw and Ed Felten[4] also present an interesting analysis of how users manage passwords, based on studies of students' behavior with real Web accounts. Their paper provides insights that those

crafting password policies would do well to consider.

The main point I want to make overall is that administrators creating password policies should be more aware of the increasing burden users have managing passwords. Administrators should construct policies with this burden to the forefront — otherwise, users will most likely use passwords in a way that counters the policy's aims.

Users in general, and Web application developers in particular, should also consider how they manage passwords — users should review the set of passwords they use, and Web application developers should consider whether and when they really require user authentication. Administrators might be able to help users by recommending tools or manual approaches for managing the many passwords involved and might also be able to encourage (or force) application developers to reduce the number of new passwords they require.

Finally, although we can't realistically expect to entirely eliminate password use today, if administrators can't define what they consider to be sufficiently secure yet usable password policy, then perhaps it really is time to move to one of the stronger authentication mechanisms the security community has been defining over the past decade.

References

1. J. Richards, A.G. Lowe-Norris, and R. Allen, *Active Directory,* 3rd ed., O'Reilly Media, 2006.
2. M. Crispin, *Internet Message Access Protocol — Version 4, Revision 1*, IETF RFC 3501, March 2003; ftp://ftp.rfc-editor.org/in-notes/rfc3501.txt.
3. J. Klensin, ed., *Simple Mail Transfer Protocol*, IETF RFC 2821, 2001; www.ietf.org/rfc/rfc2821.txt.
4. S. Gaw and E.W. Felten, "Password Management Strategies for Online Accounts," *Proc. 2nd Symp. Usable Privacy and Security* (SOUPS 06), vol. 149, ACM Press, 2006; http://doi.acm.org/10.1145/1143120.1143127.
5. W.C. Summers and E. Bosworth, "Password Policy: The Good, the Bad, and the Ugly," *Proc. Winter Int'l Symp. Information and Comm. Technologies*, ACM Int'l Conf. Proc. Series, vol. 58, 2004, pp. 1–6.
6. E. Maler and D. Reed, "The Venn of Identity: Options and Issues in Federated Identity Management," *IEEE Security & Privacy,* vol. 6, no. 2, 2008, pp. 16–23.

Stephen Farrell is a research fellow at Trinity College Dublin. His research interests include security and delay/disruption-tolerant networking. Farrell has a Joint Honors B.Sc. in mathematics and computer science from University College Dublin. Contact him at stephen.farrell@cs.tcd.ie.