DOI:10.1145/2408776.2408790

Article development led by acmqueue
queue.acm.org

Our authentication system is lacking. Is improvement possible?

BY WILLIAM CHESWICK

Rethinking Passwords

THERE IS AN authentication plague upon the land. We have to claim and assert our identity repeatedly to a host of authentication trolls, each jealously guarding an Internet service of some sort. Each troll has specific rules for passwords, and the rules vary widely and incomprehensibly.

Password length requirements vary: Dartmouth wants exactly eight characters; my broker, six to eight; Wells Fargo, eight or more. Special characters are often encouraged or required, but some characters are too special: many disallow spaces, single or double quotes, underlines, or hyphens. Some systems disallow certain characters at the beginning of the password; dictionary checks abound, including foreign language dictionaries.

Sure, brokerage, bank, and medical sites need to protect accounts from unauthorized use. So do shopping sites such as Amazon. An email account might be just as important: ask Sarah Palin. The value of an account can change over time: perhaps a new online store is added to a previously unimportant account.

Authentication may be more important to the service provider than to the client: do I care if someone gains access to my newspaper account? (The terms of use undoubtedly say I am supposed to care, but I do not.) In this case, the newspaper's very requirement of a password is a nuisance, and the password-"strengthening" rules just increase my annoyance. The marketplace does work here: studies show that competitive pressure tends to force sites toward simpler passwords.[4]

Not only do these authentication rules vary widely, the rules themselves are often considered to be part of the security secret and not available at login time, when a hint about the rules would be helpful. I call these *eye-of-newt* password rules: they remind me of the formulae for magic potions from Shakespeare. They are often particular, exacting, and sometimes difficult to satisfy. Can you think of a long passphrase that does not repeat any character more than four times?

The problem is emergent: if we had only one account, authentication would be much easier. But an active Internet user can have one- or two dozen accounts, some important, some not. These authentication trolls bother most online users, and it is easy to elicit a litany of complaints from casual users.

Many of today's rules are rooted in the deep past of security concerns, when access, threats, and targets were different. Many of these ideas were presented in the *Password Management Guideline*, (Technical Report CSC-STD-002-85), published by the Department of Defense Computer Security Center (DoD CSC) in 1985.[2] Known as the Green Book, this report was one of the Rainbow Series of books put out by the U.S. government in the 1980s and 1990s. Its advice was good at the time, and much of it still holds up, but many of our password aphorisms come from dated assumptions about threats and technology.

This is not a criticism of the original authors or their document: no sensible

security person would expect these rules to stand unamended for decades. The lore has simply not kept up with the threats and vulnerabilities.

The Green Book In Today's World

The *Password Management Guideline* came out shortly after the more-famous Orange Book (Trusted Computer System Evaluation Criteria). The Green Book was the DoD's management guideline for access to classified or sensitive government computers. It is also the basis for most of the current password rules. Most computer access at the time was either via local batch processing (with cards!) or through local or remote serial lines using terminals. The PC and Macintosh were available, but they were not especially relevant to secure computing and certainly were not networked.

Here is an important note found early in the report:

Because it is anticipated that diverse user communities will adopt this guideline, all recommendations are presented in general rather than specific terminology...Where features require the setting of a specific value (for example, password maximum lifetime), it is suggested that these be designed as parametric settings leaving the determination of exact values to local security management...

The question for today's security specialists is, what still makes sense from the 1985 guidelines? The current authentication mess suggests that we have not kept up with this task. Perhaps this article will spur some rethinking along these lines.

The DoD report offered specific advice about authentication and passwords. It stated that in a password-based authentication mechanism implemented on an ADP (automated data processing) system, passwords are vulnerable to compromise because of five essential aspects of the password system:

1. *A password must be initially assigned to a user when enrolled on the ADP system.* This rule is still fine. Many sites have used a standard password for the initial password but skipped the requirement to force a change to the default password—an attacker could simply try a number of accounts with the default password to break into a system. The same held for some reset password schemes. One solution that encouraged a change of default password was to set the default or recovery password to "I am stupid."

2. *A user's password must be changed periodically.* I will discuss this in more detail later.

3. *The ADP system must maintain a password database.* This rule is still fine.

4. *Users must remember their passwords.* It turns out this rule is unreasonable, especially for machine-generated passwords. These passwords are simply not that memorable, and to memorize multiple ones for a long time is beyond the abilities of most people. Also, people logged into many fewer systems in 1985.

5. *Users must enter their passwords into the ADP system at authentication time.* Rule 5 is incomplete: it is only single-factor authentication. The alternatives were undoubtedly well known to the authors, but probably too expensive for general deployment. I suspect that a remark to this effect at that time might have changed our world.

Furthermore, according to the report:

▸ *Users should be able to change their own passwords.* This is a good idea.

PHOTOGRAPH BY STUDIO 37

Some systems in the deep past did not allow this.

▸ *Passwords should be machine-generated rather than user-created.* It is true that machine-generated passwords tend to be much stronger: the work factor needed to crack them is easy to compute and noncontroversial. Not so for human-created passwords, where a sea of associations and language rules greatly reduces the search space.

▸ *Certain audit reports (for example, date and time of last login) should be provided by the system directly to the user.* This gives the user an opportunity to spot unauthorized accesses. The practice was widely adopted in Unix systems with the `login(1)` command. It is still a fine idea.

▸ *User ID is a unique symbol or character string that is used by an ADP system to uniquely identify a user.* The security provided by a password system should not rely on secrecy of the user's ID. This is a typical cryptographic assumption, that only the key is secret, not the user ID. (I wish this were true for the Social Security Number in the U.S.) Obscuring the user ID can be a useful barrier to wholesale attacks, however, especially against massive online systems.

▸ *Throughout the lifetime of an ADP system, each user ID should be assigned to only one person.* In other words, do not share accounts and their associated passwords. This is still a good idea for important accounts, because it may aid in logging and attribution. This can be especially important for shared accounts when a marriage is failing: former partners can be very nasty.

▸ *All user IDs should be revalidated periodically.* This is a good idea, but it is rarely implemented. Many breakins have occurred on unused or fallow accounts. Some systems implement a "we-haven't-seen-you-in-a-while" increase in authentication requirements, a good idea. A modern version includes stronger authentication when connections come from unusual locations or IP addresses.

▸ *The simplest way to recover from the compromise of a password is to change it.* Ah, the good old days! This is just wrong now. Once an account is compromised, the rot sets in and spreads through further attacks and transitive trust. Other accounts are attacked with the same password, often successfully. Bank accounts are drained (at least temporarily—personal exposure has declined on this,[3] plasma screens ordered, billing addresses changed, and identities stolen.)

On Unix/Linux personal accounts, a stolen password is just the beginning. Systems are rooted, backdoors installed, and, often, other security weaknesses are fixed. SSH (secure shell) clients are installed to capture other passwords. It tends to be easier to root a Unix host given a user account. Table 1 is a sample of the number of SUID(root) programs on a few sample operating systems, found with the command `find / -user root -perm -4000 -print`.

Table 1. Number of `setuid(root)` programs found with `find`.

System	Number of programs
Linux (Ubuntu)	19
FreeBSD 9.0	38
OSX 10.8.2	34
FreeBSD 7.2	46

Each of the examples in Table 1 is a potential root compromise, and an attacker can often find at least one.

▸ *Passwords should be changed* on a periodic basis to counter the possibility of undetected password compromise.

The most obvious threat to the security provided by a password system is from the compromise of passwords. The greater the length of time during which a password is used for authentication purposes, the more opportunities there are for exposing it. In a useful password system, the probability of compromise of a password increases during its lifetime.

This section refers to Green book Appendix C, which gets to the meat of password strength and lifetime in the face of dictionary attacks. Several simple formulae are offered (an ASCII layout and typos makes the math more difficult to follow in the online versions), and results computed for a typical case of the time.

The goal is to resist a year's worth of dictionary attacks with a cracking probability of 10^{-6} (or 10^{-20} for sensitive systems). To give one of the report's examples, a nine-character password of only uppercase letters can resist a yearlong dictionary attack over a 30-character-per-second terminal session, assuming 8.5 guesses per minute. The report offers similar computations for uppercase alphanumeric characters and words selected from a 23,300-entry dictionary of English words from four to six characters in length. The authors admit a much higher guessing rate if a file on hand is protected by a password.

Let's plug in the numbers for a modern dictionary attack using 100 million and seven billion trials per second. The first is an easy rate for a multicore machine running on typical password-hashing algorithms. The second rate is claimed for attacks implemented on modern GPUs by a commercial source. These are somewhat conservative numbers in an age of multicore processors, clusters of computers, and botnets. If you think they are too aggressive, wait a year. Table 2 shows the cracking time and password change rates for some variations.

Table 2. Cracking time and password change rates.

	Search space (bits)	7 billion trials/second		10 million trials/second	
Scheme		**Cracked in**	**Change time**	**Cracked in**	**Change time**
8-character, full alphanumeric	47.6	0.36 mins.	31.19 ms.	252.71 days	21.83 sec.
8-character, eye-of-newt	52.3	9.25 days	799.40 ms.	17.74 years	559.58 sec.
11-character, eye-of-newt	71.9	20,390 years	7 days	1.43E+07 years	14.3 years
13-character, full alphanumeric	77.4	906,123 years	331 days	2.32E+11 years	634 years
12 character, eye-of-newt	78.5	1,896,229 years	692 days	4.84E+11 years	1,327 years

The second scheme shown in the table is tougher than passwords considered secure these days: it is eight *random* characters chosen from the 93 characters found on a keyboard (a bit more than eye-of-newt). This strong password needs to be changed every *31 milliseconds* for security purposes. (My crude spreadsheet for exploring this is available on my Web site.[1])

The last two schemes in the table roughly meet the criteria of this document: the password may be changed annually without risking more than a one-in-a-million chance of compromise after a yearlong dictionary attack. These correspond to a work factor of 77–79 bits, which might surprise you as being much larger than typical password strengths actually required, usually from 20 to the mid-40s.[3] The added bits come from the requirement of 10^{-6} guessing success probability, which adds 20 bits to the password length. (The spec actually calls for a probability of 10^{-20} for classified access: that adds 66 bits!)

The one-in-a-million requirement is probably unreasonable. With an installation of very expensive brute-force hardware, I am unlikely to deploy it for a year to gain access to a high-value target if my chances of success are, say, 1%. On the other hand, history is full of examples of defenders underestimating the amount of work an attacker is willing to undertake.

Other Aphorisms

▸ *Do not use the same password on multiple services.* This is still a very good idea, though I realize that it is a pain in the neck. If I break into your Facebook account, then I am going to try that same password on LinkedIn, Gmail, iTunes, and so on. This attack works beautifully, because most people do not follow this rule.

Most practitioners who do follow this rule use a basic password, modified by some service-dependent portion. If that variable portion is obvious, they probably should not bother. In this case, it would probably be better to choose different, strong passwords and ignore the next piece of advice.

▸ *Do not write your passwords down.* This rule depends very strongly on your threat model: what are you afraid of? In the deep past, many attacks came from fellow students, co-workers, family members, and on-site spies. The movie trope of checking for Post-It notes around the desk worked, and still does.

Once an account is compromised, the rot sets in and spreads through further attacks and transitive trust. Other accounts are attacked with the same password, often successfully.

Writing your passwords down, however, is probably much safer than using the same password on multiple machines. In most cases today, the attacker does not have to be present to win. Your machine can be compromised from very far away. Or the attacker leaves infected USB thumb-sticks in the company parking lot. The check-the-Post-It attack is much less common than networked hacking attacks.

Of course, there is no need to make it too easy. Write down a comment or variation on the password that is sufficient to remind you of the real password. Sometimes I find a reminder of the particular site's eye-of-newt rule is enough.

Password wallets are a terrific idea for storing passwords, but they get you back in the game of storing secrets on possibly unsecured computers with network access. The yellow pad in your office is probably more secure.

▸ *Change your passwords often.* This is often enforced by the authentication service, and it is generally a bad idea—and not useful. A good, strong password that you can remember is difficult to create and probably difficult to remember, especially if there are different passwords for different accounts. When a password is changed by force, all that goodness goes away, requiring a whole new effort.

This can be a particular problem for rarely used passwords. For example, corporate-provided health care in the U.S. requires employees to review and make changes to coverage options annually. These systems require strong authentication and tend to be used exactly once a year, so to remember the password at all, I either write it down or rely on the password-recovery scheme. On some systems, I have cycled through several strong passwords over a longer period than the authentication server remembers. Those really good passwords are too good to let go.

What We Have Learned

It is simply poor engineering to expect people to choose and remember passwords that are resistant to dictionary attacks. User training does not work:

people will write down their passwords regardless.

Fortunately, dictionary attacks are rarely the problem. They are completely frustrated by getting out of the game: limit the number of attempts to a handful, then disable the account. Multiple-factor authentication and better recovery from compromise have also helped out.

This is not a new idea. I got my first bank ATM card in the early 1970s; it had a four-digit PIN. I do not recall if I was allowed to select the PIN myself, but it did not matter: it was my only PIN, and the service was unique and useful enough that I committed the PIN to memory. If I forgot it, the card would be eaten, or the account locked. This policy is still used in the U.S. banking system some four decades later, proof that it is working. It is also not a rare solution. Most authentication systems lock a user out after several tries.

More importantly, the threats have changed. Dictionary attacks on passwords are not nearly the problem they used to be. Today's threats include:

- Keystroke loggers record any password, no matter how complex.
- Phishing sites capture the passwords of the unwary, and it is very easy to be unwary. The mail reader should present any URL found in an email with red flags and warnings, especially if it refers to an unfamiliar domain.
- Password files from poorly protected servers spill our secrets across the Internet, eye-of-newt passwords included.
- Sites that have passed state-of-the-art security audits are later found to have been leaking credit card information for years. Best-in-practice may be good enough for the lawyers, but it really is not solving very hard security problems.

Client systems are hardly secure—we have built our houses on sand. Why should any mouse click present a security threat?

Dictionary attacks can be launched on password wallets, SSH agent passphrases, PGP (Pretty Good Privacy) key rings, and stolen password databases. For strong passwords, words, rather than eye-of-newt strings, are easier to type and remember. From the Brown corpus's top 23,300 common English words, I generated several random passphrases in the spirit of STD-002 and xkcd:[5]

- fooled otherwise faustus
- exclaimed democrat cruz
- deauville attaches ornamented
- acutely jeep pasha

These give a search space of more than 43 bits, matching the estimated strength of today's strongest passwords. They also offer a chance to expand one's vocabulary. Alas, they probably fail most eye-of-newt rules.

Suggestions

My dream is that authentication might become a lot less odious, maybe even fun. Passwords and passphrases should be easier to type and include automatic correction for typing and "tapographical" errors (on smart phones). This can be done without loss of security.

Why do the eye-of-newt rules remain? Account unlocking is a problem, requiring relatively costly or unsecured secondary authentication efforts. In some cases, it would be appropriate to have someone else—for example, an authorized spouse on a shared back account—enable the temporary authentication and subsequent password change. "Honey, I did it again" could be much easier than getting through to an 800 number on a weekend.

It would be nice to have more than one way to log into a site, each way having about the same strength. This gives the users a choice of authentication methods, with other methods as a backup login. (Mother's maiden name is *not* what I am talking about here. Secondary passwords tend to be much weaker and should not be used. Security history is full of attacks that force the defender to drop back to secondary, less effective defenses.)

If one tries the same password twice in a session, that should not count as two tries. We all make, or suspect that we make, typographical errors. Did I enter that password correctly? I will try again more carefully. This should not count as a second wish for the password troll.

Conclusion

I am not optimistic that these changes will happen rapidly, or even at all. There is a huge installed base out there. "We do the same thing as everybody else" is an effective legal defense against malfeasance, so why change things? (I hate the word *legacy*!)

Authentication systems are vital, and changes to them can produce widespread and embarrassing failures. It is not clear that easier authentication would provide a market advantage. Is a company less secure than another company because it is easier to log into? Will it gain market share by doing so?

In spite of all this, the system seems to be working. We are leaking military and industrial secrets to attackers all over the world, but millions of people use the Internet successfully every day, and it is an important part of the world's economy. Somehow, we get by.

Finally, I would like to see these systems engineered such that the user needs to remember only one security maxim: *Don't be a moron.* Do not pick a password that someone who knows you can guess in a few tries, or that someone watching you type can figure out easily.

Unlike the eye-of-newt password rules, this last rule makes sense to the casual user and is easy to remember. All we have to do is engineer the rest to be reasonably secure. C

Related articles on queue.acm.org

Security - Problem Solved?
John Viega
http://queue.acm.org/detail.cfm?id=1071728

Building Secure Web Applications
George V. Neville-Neil
http://queue.acm.org/detail.cfm?id=1281889

LinkedIn Password Leak: Salt Their Hide
Poul-Henning Kamp
http://queue.acm.org/detail.cfm?id=2254400

References

1. Cheswick, W. 2012; http://www.cheswick.com/ches/papers/std-002-results.xls; and http://www.cheswick.com/ches/papers/std-002-results.numbers.
2. Department of Defense Computer Security Center. *Password Management Guideline*, 1985.Technical Report CSC-STD-002-85.
3. Florêncio, D. and Herley, C. Is everything we know about password stealing wrong? *IEEE Security and Privacy 99* (2012). DOI 10.1109/MSP.2012.57.
4. Florêncio, D. and Herley, C. Where do security policies come from?. In *Proceedings of the Sixth Symposium on Usable Privacy and Security* (2012). ACM, NY, DOI 10.1145/1837110.1837124. http://doi.acm.org/10.1145/1837110.1837124.
5. xkcd; http://xkcd.com/936/.

William "Ches" Cheswick was formerly with Bell Labs, Lumeta Corporation, and AT&T Shannon Lab. One of his current projects is promoting better passwords. The earliest password he can remember using is "polpis," a location in Nantucket. He now uses multiple random words for important accounts, and writes stuff down.